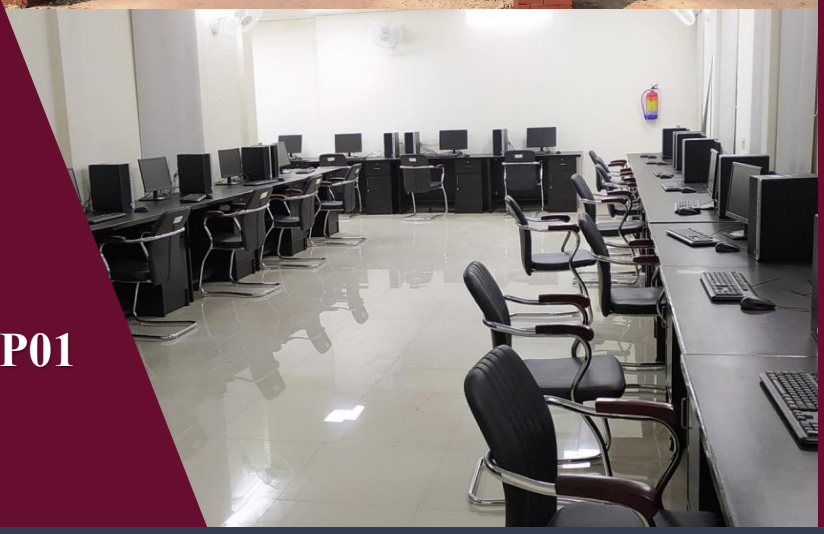
2025-26

جب کوئی قوم فن اور علم
سے عاری ہو جاتی ہے
تو وہ غربت کو دعوت
دیتی ہے اور جب غربت
آتی ہے تو وہ ہزاروں
جرائم کو جنم دیتی ہے۔

"When a nation becomes devoid of art and learning, it invites poverty and when poverty comes it brings in its wake thousands of crimes."
-Sir Syed Ahmad Khan



B.Sc. (CA) III Semester Lab Manual

Lab-III

Course Code- CABSMJ3P01

DEPARTMENT OF COMPUTER SCIENCE

ALIGARH MUSLIM UNIVERSITY, ALIGARH

2025-2026

Credits

The following lab manual up-gradation committee updated the lab manual:

- ❒ *Prof. Arman Rasool Faridi (Chairperson)*
- ❒ *Prof. Mohammad UbaidUllah Bokhari*
- ❒ *Prof. Aasim Zafar*
- ❒ *Prof. Suhel Mustajab*
- ❒ *Dr. Shafiqul Abidin*
- ❒ *Dr. Asif Irshad Khan*
- ❒ *Dr. Faisal Anwar*
- ❒ *Dr. Mohammad Sajid*
- ❒ *Dr. Mohammad Nadeem*
- ❒ *Dr. Faraz Masood*
- ❒ *Dr. Mohammad Luqman*

The following committee member originally design the lab manual:

- ❒ *Prof. Mohammad Ubaidullah Bokhari*
- ❒ *Prof. Arman Rasool Faridi*
- ❒ *Dr. Faisal Anwer*
- ❒ *Prof. Aasim Zafar (Convener)*

Design & Compilation:

- ❒ *Dr. Faraz Masood*
- ❒ *Dr. Mohammad Luqman*

Revised Edition: July, 2025

Department of Computer Science, A.M.U., Aligarh (U.P.) India

COURSE TITLE: Lab -III

COURSE CODE: CABMSJ3P01

CREDIT: 2

PERIODS PER WEEK: 3

CONTINUOUS ASSESSMENT: 40

EXAMS: 60

Lab Manual: Computer Lab – CABSMJ3P01

LIST OF CONTENTS (WEEK WISE)

APPENDIX–I

Template for the Index of Lab File

WEEK NO.	PROBLEMS WITH DESCRIPTION		PAGE NO.	SIGNATURE OF THE TEACHER WITH DATE
1	1#			
	2#			
	3#			
2	1#			
	2#			
	3#			
3	1#			
	2#			
	3#			

Note: The students should use Header and Footer mentioning their roll no. & name in footer and page no in header.

INTRODUCTION

This assignment on Java is designed for the students of B.Sc. (Computer Applications) -III Semester. Java is one of the most popular object-oriented programming languages. Its applications are enormous and a good understanding of Java will definitely give the students a technical edge. Many open-source editors of Java are available. Some famous of them are Eclipse, NetBeans, BlueJ and JCreator.

OBJECTIVES

After completing this Lab assignment, the students should be able to:

- To develop an understanding of basic concepts of java.
- To get familiarize with Java data types, branching and looping construct, Class, Objects, Inheritance, Abstract Class, Polymorphism, Interface etc.

LAB INSTRUCTIONS

- The students need to submit the solutions of each exercise.
- The students need to ensure that each question is assessed and signed by the Teacher.
- All Students are required to complete the assignment before due date.
- Late submission would not be accepted.
- Cooperate, collaborate and explore for the best individual learning outcomes but **copying is strictly prohibited**.

WEEK #1

Write a short report (1–2 pages) on **"Resources for Learning and Practicing Java Programming"**. Your report should include:

1. **Official Documentation and Websites**

   - Identify official sources (e.g., Oracle Java documentation, OpenJDK, etc.).
   - Mention what type of information these sites provide (tutorials, API references, downloads).

2. **Books and E-Books**

   - List at least **2 recommended books or e-books** for beginners.
   - Briefly explain why they are useful.

3. **Online Learning Platforms**

   - Mention **3 online platforms** (free or paid) where Java programming is taught (e.g., Codecademy, Coursera, Udemy).
   - Write 2–3 lines about what kind of courses or practice exercises they offer.

4. **Coding Practice Websites**

   - List **3 websites or apps** where you can practice Java problems (e.g., HackerRank, LeetCode, Codeforces).

5. **Community & Discussion Forums**

   - Include **forums or communities** where Java programmers discuss and solve problems (e.g., Stack Overflow, Reddit r/java).

6. **Your Preferred Resources**

   - Write 3–4 sentences about **which resources you personally find most useful** and why.

WEEK #2

Introduction and software requirements for HTML/Javascript PHP programs:

1# What are the softwares that helps to run java programs.

2# What is JDK and JRE.

3# What is eclipse IDE.

4# How to run the java program in eclipse/netbeans IDE.

WEEK #3

Basics problems in java

1# Write a java program to add the two numbers.

2# Write a java program to multiply two floating numbers.

3# Write a java program to display a cube of a number .

4# Write a Java program that takes three numbers as input to calculate and print the average of the numbers.

5# Write a Java program to compute the distance between two points.

Optional

6# Write a Java program to swap two numbers using a temporary variable.

7# Write a Java program to calculate the area of a rectangle given its length and breadth.

8# Write a Java program to convert temperature from Celsius to Fahrenheit.

9# Write a Java program that takes two integer inputs and computes their remainder and quotient.

10# Write a Java program to find the circumference of a circle given its radius.

WEEK #4

Problems Based on if statement/Looping in JAVA

1# Write a java program to check whether the given number is odd or even.

2# Write a java program to find the largest number among the three numbers.

3# Write a Java program that takes a number as input and prints its multiplication table upto 10.

4# Write a Java program to calculate the sum of following series:

1 + 2 + 3 + 4 + ... + N

5# Write a Java program to take a number, divide it by 2 and print the result until the number becomes less than 10.

Optional:

6# Write a Java program to check whether a given character is a vowel or consonant.

7# Write a Java program to find the smallest number among four given numbers.

8# Write a Java program to calculate the sum of all even numbers from 1 up to a given number N.

9# Write a Java program to check whether a given year is a leap year or not.

10# Write a Java program that takes a number as input and prints all its factors.

WEEK #5

Problems Based on Array in JAVA

1# Write a Java program to insert 10, 20, 30 ....in an array and display them.

2# Write a Java program to calculate the sum of all the array elements.

3# Write a java program to print the following pattern.

```
    1
   12
  123
 1234
12345
```

4# Write a java program to find the sum of following series where n is input by the user.

1+1/2+1/3+1/4+.....+1/n.

5# Write a Java program and compute the sum of the digits of an integer.

6# Write a Java program to calculate the factorial of a number.

Optional:

7# Write a Java program to find the largest element in a given integer array.

8# Write a Java program to reverse the digits of a given integer.

9# Write a Java program to check if a given number is a palindrome or not.

10# Write a Java program to convert a decimal number into Hexadecimal number and vice-versa.

11# Write a Java program to print the following pattern:

```
  *
 **
***
 **
  *
```

WEEK #6

Problems Based on If statement/Looping/Array in JAVA

1# Write a Java program to print the odd numbers from 1 to 99.

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

3# Write a Java program to swap the first and last elements of an array.

4# Write a Java program to find the maximum and minimum among array elements.

5# Write a Java program to print all prime numbers between 0 to 100.

6# Write a Java program to implement linear search.

Optional:

7# Write a Java program to print all prime numbers between 0 to 100.

8# Write a Java program to find the second largest element in an array.

9# Write a program to implement Fibonacci series up to N terms (0,1,1,2,3,5....).

10# Write a Java program to reverse all elements of an array.

11# Write a Java program to find the frequency of each character in a given string.

WEEK #7

Problems Based on If statement/Looping/Array/Strings in JAVA

1# Write a Java function to implement binary search.

2# Write a Java function to arrange the elements of an array in ascending order (Sorting).

3# Write a program to reverse a given string.

4# Write a program to check whether a given string is palindrome or not.

5# Write a program to implement factorial of a number through recursion.

6# Write a program to implement Fibonacci series of a number with and without recursion.

Optional:

7# Write a Java function to find the greatest common divisor (GCD) of two numbers with and without using recursion.

8# Write a program to check whether two strings are anagrams of each other ("listen" and "silent" are anagrams).

9# Implement quick sort using recursion.

WEEK #8

Problems Based on Object / Class / Constructor in JAVA

1# Create a class FRUIT which has data members color, taste and price. Also create a method display() which will print values of FRUIT object. Create three objects of FRUIT class and call their display() methods.

2# Create a class FRUIT which has data members color, taste and price. It has a method setDetails() which will set the values of color, taste and price. Also create a method display() which will print values of FRUIT object.

3# In previous question, set the values of using color, taste and price using Constructor.

4# Add one-argument constructor and two-argument constructor in addition to default constructor in FRUIT class.

5# Use the concept of constructor-chaining in the previous question using this().

Optional

6# Create a class CAR with the following details:

Data members: model, color, price.

Member methods:

setDetails() – to set values of all data members using setters.

getDetails() – to get values of all data members using getters.

display() – to print all details of the car.

Requirements:

1. Implement default constructor to initialize default values.
2. Implement a parameterized constructor (with one argument) to set only model.
3. Implement another parameterized constructor (with two arguments) to set model and color.
4. Use constructor chaining to reduce code redundancy.
5. Create three objects of CAR class using:
    - Default constructor
    - One-argument constructor
    - Two-argument constructor
6. Set price for each object using the setDetails() method.
7. Call the display() method for each object.

WEEK #9

Problems Based on Inheritance in JAVA:

1# Create a class Vehicle, write a method cost() in this class. Create two classes Bus and Train which have their own display() methods and inherit from Vehicle class. Create objects of Bus and Train class and call cost() and display() methods.

2# Create class University which has data members- name and ranking. Create class Faculty that extends University class has data member- name and method- Details(). Create a new class Department which is derived from Faculty and has data member- name, chairman and method- Details() and Display() where Display() method calls Details() methods of both Faculty and Department class in its body. Create an object of Department class to Display() method and University ranking.

Problems Based on Static methods in JAVA

3# Create class Account (Data members- Id, Account_holder_name, Address; Methodsdeposit(), withdraw()). Create two static methods in Account calculateSimpleInterest() and calculateCompoundInterest() and implement them.

Problems Based on Abstract class in JAVA

4# Create class Account (Data members- Id, Account_holder_name, Address; Methodsdeposit(), withdraw()). Declare deposit() and withdraw() as abstract methods. Declare Account class as abstract. (Create constructor in Account as well).

5# Create two children of Account- Saving (Data Members- Min_balance; Methodsdisplay(), deposit(), withdraw()) and Current (Data Members- Max_withdrawl_limit; Methods-display(),deposit(), withdraw()) . Create constructors for both classes. Implementation of deposit() and withdraw() should be specific to Saving and Current class. Create objects of Saving and Current class and display them.

Optional:

6# Create a class Shape with a method area(). Create two derived classes Rectangle and Circle that extend Shape. Each class should override the area() method to calculate the area of the respective shape. Create objects of Rectangle and Circle and call their area() methods.

7# Create a class Employee with data members: name, salary, and a method showDetails(). Create a class Manager that extends Employee with an additional data member department. Override the showDetails() method in Manager to display all details, including department. Create an object of Manager and call showDetails().

8# Create an abstract class Appliance with data members brand, power and abstract methods turnOn() and turnOff(). Create two derived classes WashingMachine and Refrigerator that provide their own implementations of turnOn() and turnOff(). Create objects of WashingMachine and Refrigerator and call their methods.

9# Create a class MathOperations with two static methods: findGCD(int a, int b) to calculate the greatest common divisor and findLCM(int a, int b) to calculate the least common multiple. Call these methods without creating an object of MathOperations.

10# Create a class Student with data members rollNo, name, marks. Add a static variable schoolName. Create static method changeSchoolName() to update schoolName. Demonstrate how the static variable is shared among all objects.

WEEK #10

Problems Based on Nested Class in JAVA

1# Create class Person (Data Member- name, phone). Create two member inner classes Address (Data Member- House_No, Street, City, State; Method- displayAddr()) and DateOfBirth (Data Member- Day, Month,Year; Method- displayDOB()). Display() is the method of Person class which will display name, address and date of birth of a Person object.

2# Create class Edible. Within that define two static classes Fruit and Vegetable. Fruit class will have two methods- fruitDetails() is a static method and fruitPackaging() is a non-static method. Vegetable class also has similar methods - vegetableDetails() and vegetablePackaging(). Call all the four methods from main method.

Problems Based on Static Polymorphism in JAVA:

3# Create three different minMaxAdd() methods to calculate minimum, maximum and addition of integers, real numbers and characters.

Problems Based on Dynamic Polymorphism in JAVA:

4# Create a class **ObjectOriented** which has methods- abstraction(), polymorphism() and inheritance(). Create a class **JavaLanguage** which inherits from ObjectOriented class and has its own methods- persistence() and interfaces(). Create an object of **JavaLanguage** class to access all of its own and parent's methods.

5# In previous question, create a new class **C++** which also inherits from **ObjectOriented** class and has its own methods- template() and

friendFunction().Create an object of **C++** class to access all of its own and parent's methods.

6# Create class **University** which has data members- name and ranking. Create class **Faculty** that extends **University** class has data member- name and method- Details(). Create a new class **Department** which is derived from **Faculty** and has data member- name, chairman and method- Details() and Display() where Display() method calls Details() methods of both **Faculty** and **Department** class in its body. Create an object of **Department** class to Display() method and University ranking.

Optional:

7# Create a class Employee (Data Members – empName, empId). Create two member inner classes:

- Salary (Data Members – basic, hra, pf; Method – displaySalary() to print salary details)
- JoiningDate (Data Members – day, month, year; Method – displayJoiningDate() to print joining date).

In the Employee class, create a method displayEmployee() that prints the employee's name, ID, salary details, and joining date.

8# Create a class Shape with overloaded methods area():

- area(int side) – calculates area of a square.
- area(int length, int breadth) – calculates area of a rectangle.
- area(double radius) – calculates area of a circle.

9# Create a class Vehicle with a method run(). Create subclasses Bike and Car that override the run() method. In the main() method, use a reference of Vehicle to call run() for objects of Bike and Car.

WEEK #11

Problems Based on Interface in JAVA:

1# Create an interface **Account** having methods- deposit(), withdraw() and aboutBank() (abountBank() is a static method). Create two classes **Saving** and **Current** which implement the **Account** interface. Call the methods of **Saving** and **Current** classes in main method.

2# In the previous question, create a new method in **Account** interface- takeLoan() (takeLoan() is a default method). takeLoan() method would be implemented by **Saving** class only. Call the methods of **Saving** and **Current** classes in main method.

3# Create interfaces **Bike** and **Scooty**, both of which have two methods- offer() and details() (details() is default method). Create a new class **BuySomething** which implements both interfaces. To remove ambiguity, create a method details() in **BuySomething** class as well in which call the details() method of both interfaces. Call the methods of **BuySomething** class in main method.

Optional:

4# Create two interfaces Printer and Scanner, both having methods connect() and details() (details() is a default method). Create a class MultiFunctionMachine that implements both interfaces. In MultiFunctionMachine, override the details() method to resolve ambiguity and call the details() methods of both interfaces. Call all methods of MultiFunctionMachine in the main() method.

5# Create an interface Device with a method powerOn(). Create another interface SmartDevice that extends Device and adds a method connectWiFi(). Create a class SmartPhone that implements SmartDevice.

Demonstrate calling both powerOn() and connectWiFi() using a SmartPhone object in the main() method.

WEEK #12

Problems Based on Exception Handling in JAVA:

1# Write a program that calls a method that throws an exception of type ArithmeticException in a for loop at an undesirable situation (such as divide by zero or taking square root of negative number). Catch the exception and display appropriate message. (Example of Unhecked Exception).

2# Write a program of your choice where a Checked Exception occurs at third function but handled at the first calling function. Use both ways of managing Checked Exception i.e. using try-catch block and throws keyword.

3# You are developing an online banking system where users can transfer money between accounts. If a user tries to withdraw more money than is available in their account, an InsufficientFundsException should be thrown.

4# Create a user-defined exception InvalidAgeException when the age of a person is below 18 years. Use this exception at appropriate place.

WEEK #13

Problems Based on File Handling in JAVA:

1# Write a Java Program to create a new file.

2# Write a Java Program to write into a file.

3# Write a Java Program to copy one file into another file.

4# Write a java program to find total no. of characters in a file.

5# Write a java program to find total no. of lines in a file.

WEEK #14

Problems Based on JDBC API in JAVA:

1# Write a Java program to:

a. Connect with a database of your choice using JDBC API.

b. Create an Employee table having employee id, age, name and salary.

c. Insert five records in to Employee table.

d. Delete any two records.