



19CSE204 – OBJECT ORIENTED PARADIGM

LAB PRACTICALS 1

SUBMITTED BY: MADHUNISHA M V

CH.EN.U4CCE22017

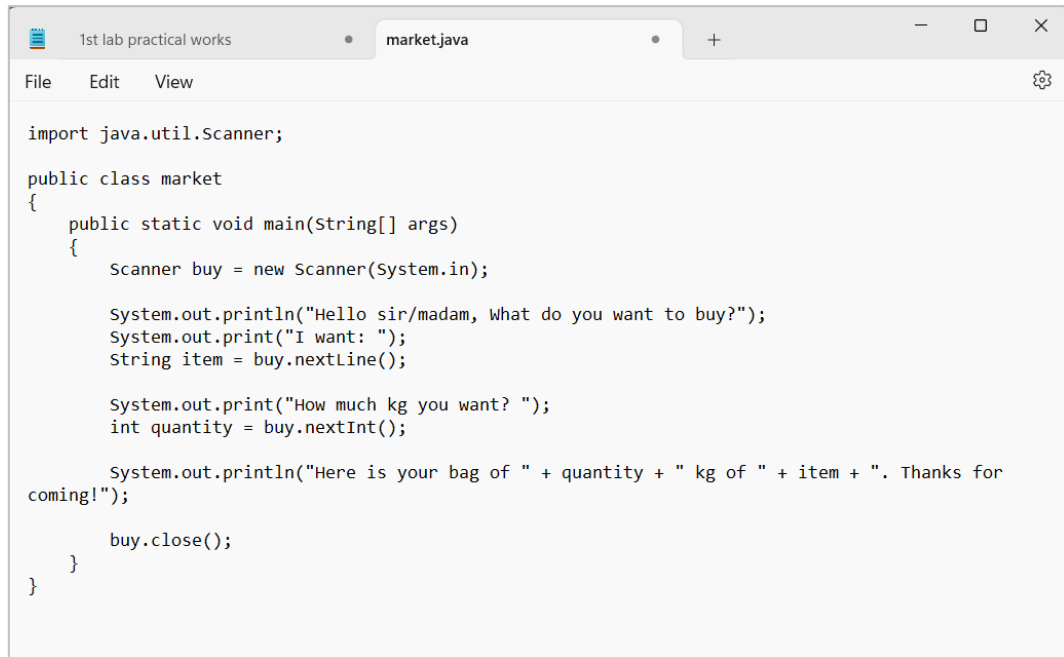
SUBMITTED TO: DR. S. SUTHIR

SUBMITTED ON: 23.07.2025

1) Input Output Statement Combinations (With Scanner)

Objective: To develop a simple Java application using the Scanner class for handling user input and generating appropriate output messages.

Code:

A screenshot of a code editor window titled '1st lab practical works' with a tab for 'market.java'. The code is as follows:

```
import java.util.Scanner;

public class market
{
    public static void main(String[] args)
    {
        Scanner buy = new Scanner(System.in);

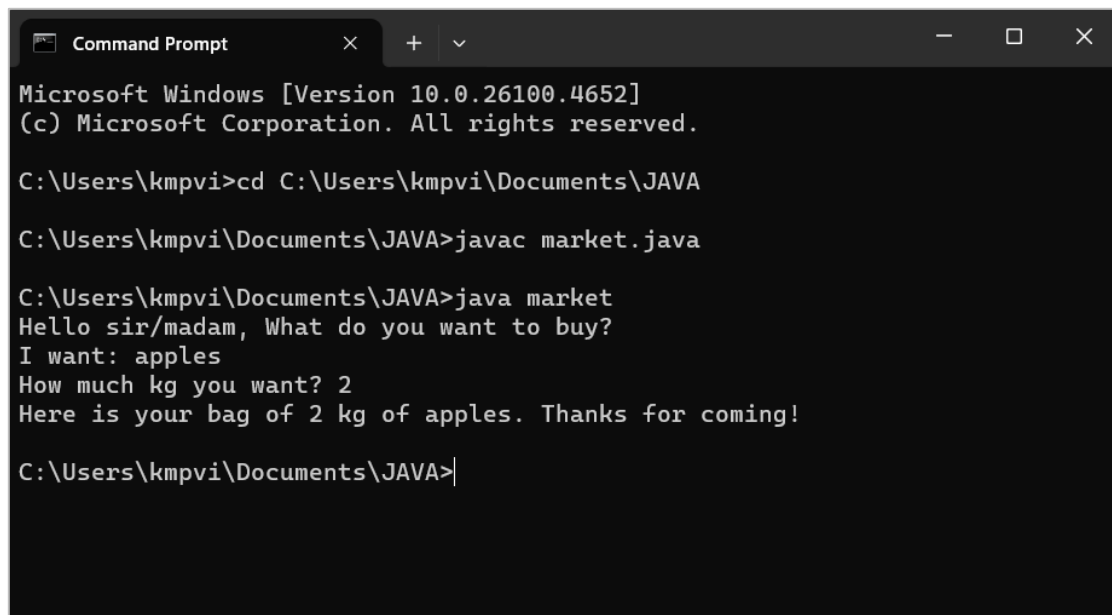
        System.out.println("Hello sir/madam, What do you want to buy?");
        System.out.print("I want: ");
        String item = buy.nextLine();

        System.out.print("How much kg you want? ");
        int quantity = buy.nextInt();

        System.out.println("Here is your bag of " + quantity + " kg of " + item + ". Thanks for coming!");

        buy.close();
    }
}
```

Output:

A screenshot of a Windows Command Prompt window. The text shown is:

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kmpvi>cd C:\Users\kmpvi\Documents\JAVA

C:\Users\kmpvi\Documents\JAVA>javac market.java

C:\Users\kmpvi\Documents\JAVA>java market
Hello sir/madam, What do you want to buy?
I want: apples
How much kg you want? 2
Here is your bag of 2 kg of apples. Thanks for coming!

C:\Users\kmpvi\Documents\JAVA>|
```

This program helps us understand how to take input from the user in Java using the Scanner class. It shows how to read a string (item name) and an integer (quantity) from the console. The program then combines these inputs and displays a proper output message to the user. It explains how input and output statements work together for user interaction in a real-world situation like buying something in a market.

2) All Data Operations

Objective: To create a Java program that demonstrates the use of various data operators (arithmetic, unary, relational, and logical).

Code:

```
1st lab practical works • VotingEligibility.java × + − □ ×
File Edit View ⚙

import java.util.Scanner;

public class VotingEligibility {
    public static void main(String[] args) {
        Scanner vote = new Scanner(System.in);

        System.out.println("--Voting Eligibility Checker --");

        // Arithmetic Operators
        System.out.print("Enter your current age: ");
        int age = vote.nextInt();

        System.out.print("Enter years you want to add to age: ");
        int addYears = vote.nextInt();

        int newAge = age + addYears;           // + addition
        int ageDifference = newAge - 18;       // - subtraction
        int doubleAge = newAge * 2;           // * multiplication
        int halfAge = newAge / 2;             // / division
        int remainder = newAge % 2;           // % modulus

        System.out.println("\n Arithmetic Results ");
        System.out.println("Future Age (+): " + newAge);
        System.out.println("Difference from 18 (-): " + ageDifference);
        System.out.println("Double Age (*): " + doubleAge);
        System.out.println("Half Age (/): " + halfAge);
        System.out.println("Remainder when divided by 2 (%): " + remainder);

        // Unary Operators
        System.out.println("\n Unary Results ");
        System.out.println("Current Age++ (Increment): " + (++age));
        System.out.println("Future Age-- (Decrement): " + (--age));

Ln 7, Col 57 100% Windows (CRLF) UTF-8
```

```
1st lab practical works • VotingEligibility.java × + − □ ×
File Edit View ⚙

        // Unary Operators
        System.out.println("\n Unary Results ");
        System.out.println("Current Age++ (Increment): " + (++age));
        System.out.println("Future Age-- (Decrement): " + (--age));

        // Relational Operators
        System.out.println("\n Relational Results ");
        System.out.println("Is your age < 18? " + (age < 18));
        System.out.println("Is your age >= 18? " + (age >= 18));
        System.out.println("Is your age == 18? " + (age == 18));
        System.out.println("Is your age != 18? " + (age != 18));

        // Logical Operators
        System.out.println("\n Logical Results ");
        boolean hasVoterID = true;
        boolean isCitizen = true;

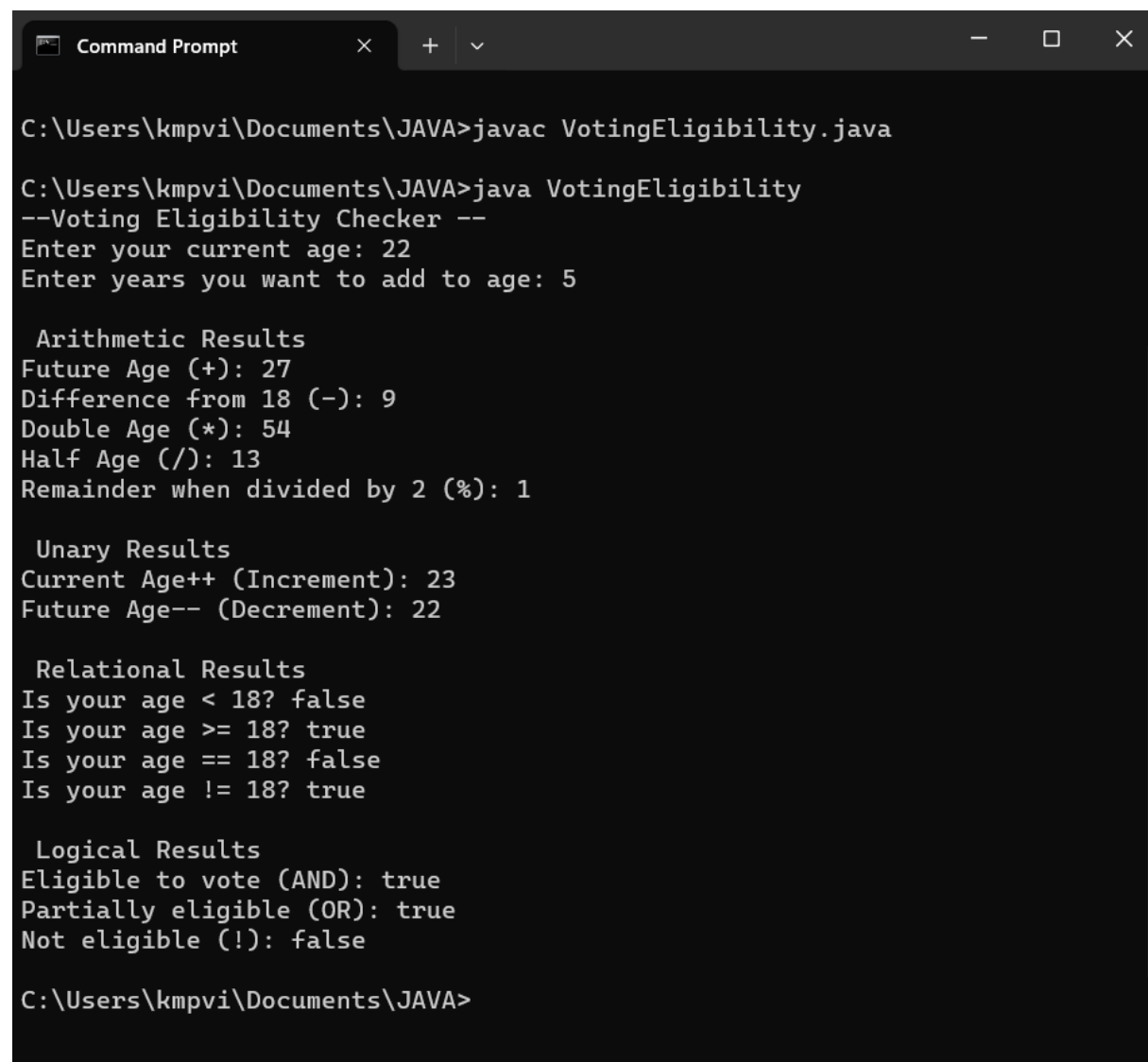
        boolean eligible = (age >= 18) && hasVoterID && isCitizen; // AND
        boolean partiallyEligible = (age >= 18) || hasVoterID;      // OR
        boolean notEligible = !(age >= 18);                       // NOT

        System.out.println("Eligible to vote (AND): " + eligible);
        System.out.println("Partially eligible (OR): " + partiallyEligible);
        System.out.println("Not eligible (!): " + notEligible);

        vote.close();
    }
}

Ln 7, Col 57 100% Windows (CRLF) UTF-8
```

Output:



```
C:\Users\kmpvi\Documents\JAVA>javac VotingEligibility.java

C:\Users\kmpvi\Documents\JAVA>java VotingEligibility
--Voting Eligibility Checker --
Enter your current age: 22
Enter years you want to add to age: 5

  Arithmetic Results
Future Age (+): 27
Difference from 18 (-): 9
Double Age (*): 54
Half Age (/): 13
Remainder when divided by 2 (%): 1

  Unary Results
Current Age++ (Increment): 23
Future Age-- (Decrement): 22

  Relational Results
Is your age < 18? false
Is your age >= 18? true
Is your age == 18? false
Is your age != 18? true

  Logical Results
Eligible to vote (AND): true
Partially eligible (OR): true
Not eligible (!): false

C:\Users\kmpvi\Documents\JAVA>
```

This program helps us understand how to perform different types of operations in Java while checking voting eligibility. It uses the **Scanner class** to take the user's age and an additional number of years as input. The program demonstrates:

- **Arithmetic operators** like +, -, *, /, and % to calculate future age, age difference, and other values.
- **Unary operators** (++ and --) to increment and decrement age.
- **Relational operators** (<, >=, ==, !=) to compare age with the voting eligibility age (18 years).
- **Logical operators** (&&, ||, !) to check conditions like age, citizenship, and voter ID to determine voting eligibility.

3) All Datatypes

Objective: To develop a Java program that demonstrates the use of various primitive data types (byte, short, int, long, float, double, char, and Boolean).

Code:

```
1st lab practical works • StudentInfo.java × ModifiersApplication.java +
File Edit View
import java.util.Scanner;

public class StudentInfo {
    public static void main(String[] args) {
        Scanner student = new Scanner(System.in);

        System.out.println("-- Student Information System --");

        // Integer types
        byte rollNumber; // 1 byte
        short totalSubjects; // 2 bytes
        int age; // 4 bytes
        long contactNumber; // 8 bytes

        // Floating-point types
        float height; // 4 bytes
        double gpa; // 8 bytes

        // Character type
        char grade; // 2 bytes

        // Boolean type
        boolean isPassed; // 1 byte

        System.out.print("Enter Roll Number (byte): ");
        rollNumber = student.nextByte();

        System.out.print("Enter Total Subjects (short): ");
        totalSubjects = student.nextShort();

        System.out.print("Enter Age (int): ");
        age = student.nextInt();

        Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

```
1st lab practical works • StudentInfo.java × ModifiersApplication.java +
File Edit View
        System.out.print("Enter Age (int): ");
        age = student.nextInt();

        System.out.print("Enter Contact Number (long): ");
        contactNumber = student.nextLong();

        System.out.print("Enter Height in meters (float): ");
        height = student.nextFloat();

        System.out.print("Enter GPA (double): ");
        gpa = student.nextDouble();

        System.out.print("Enter Grade (char): ");
        grade = student.next().charAt(0);

        System.out.print("Has the student passed? (true/false): ");
        isPassed = student.nextBoolean();

        System.out.println("\n----- Student Details -----");
        System.out.println("Roll Number : " + rollNumber);
        System.out.println("Total Subjects : " + totalSubjects);
        System.out.println("Age : " + age);
        System.out.println("Contact Number : " + contactNumber);
        System.out.println("Height : " + height + " meters");
        System.out.println("GPA : " + gpa);
        System.out.println("Grade : " + grade);
        System.out.println("Passed : " + isPassed);

        student.close();
    }
}

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Output:

```
Command Prompt

C:\Users\kmpvi\Documents\JAVA>javac StudentInfo.java

C:\Users\kmpvi\Documents\JAVA>java StudentInfo
-- Student Information System --
Enter Roll Number (byte): 17
Enter Total Subjects (short): 6
Enter Age (int): 22
Enter Contact Number (long): 7200272900
Enter Height in meters (float): 151
Enter GPA (double): 8
Enter Grade (char): A
Has the student passed? (true/false): true

----- Student Details -----
Roll Number      : 17
Total Subjects   : 6
Age              : 22
Contact Number   : 7200272900
Height           : 151.0 meters
GPA              : 8.0
Grade            : A
Passed           : true

C:\Users\kmpvi\Documents\JAVA>
```

This program helps us learn how to use different data types in Java to store student information. It accepts various student details:

- **byte** for roll number,
- **short** for total subjects,
- **int** for age,
- **long** for contact number,
- **float** for height,
- **double** for GPA,
- **char** for grade, and
- **boolean** for pass/fail status.

The program then displays all the details in an organized way. This shows how input and output work together and why using the correct data type for each kind of information is important in real-life applications like managing student records.

4) All Access Modifiers

Objective: To develop a Java program that demonstrates the use of different access modifiers (public, private, protected, and default) and non-access modifiers (abstract, final, static) to understand their scope, visibility, and usage in object-oriented programming.

code:

```
1st lab practical works x ModifiersApplication.java x +
File Edit View

// Abstract Class (Non-Access Modifier)
abstract class LibraryItem {
    // Protected variable (Access Modifier)
    protected String title;

    // Constructor
    public LibraryItem(String title) {
        this.title = title;
    }

    // Abstract method (Non-Access Modifier)
    abstract void displayDetails();

    // Public method (Access Modifier)
    public void showTitle() {
        System.out.println("Title: " + title);
    }
}

// Class with Access Modifiers
public class ModifiersApplication extends LibraryItem {
    // Public variable
    public String author;

    // Private variable (can only be accessed inside this class)
    private int itemID;

    // Default variable (no modifier: accessible in same package)
    int publishedYear;
}

Ln 73, Col 83 100% Windows (CRLF) UTF-8
```

```
1st lab practical works • ModifiersApplication.java x +
File Edit View

// Default variable (no modifier: accessible in same package)
int publishedYear;

// Static variable (Non-Access Modifier)
static String libraryName = "City Library";

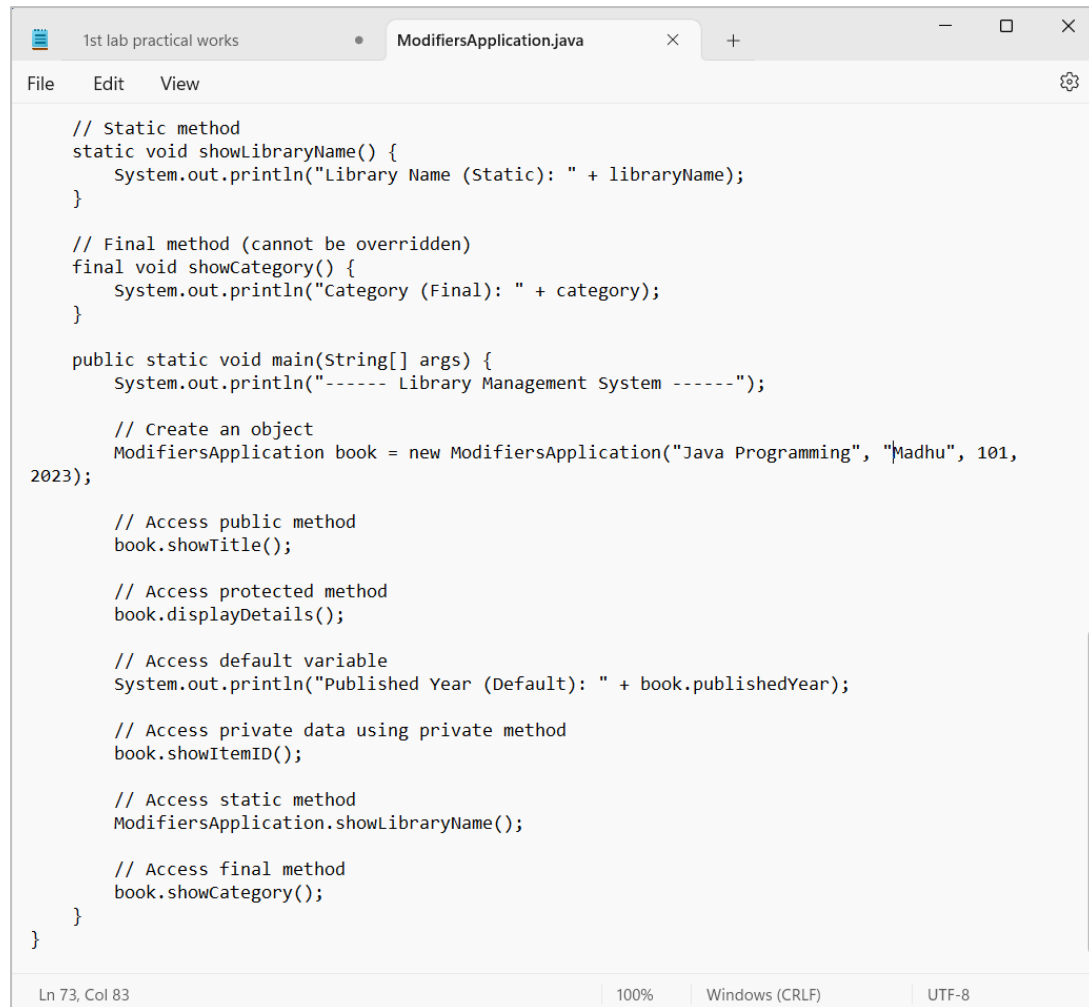
// Final variable (Non-Access Modifier)
final String category = "Books";

// Constructor
public ModifiersApplication(String title, String author, int itemID, int publishedYear) {
    super(title); // Call abstract class constructor
    this.author = author;
    this.itemID = itemID;
    this.publishedYear = publishedYear;
}

// Implement abstract method
@Override
void displayDetails() {
    System.out.println("Item ID : " + itemID);
    System.out.println("Author : " + author);
    System.out.println("Published Year : " + publishedYear);
    System.out.println("Category : " + category);
}

// Private method
private void showItemID() {
    System.out.println("Accessing private itemID: " + itemID);
}

Ln 73, Col 83 100% Windows (CRLF) UTF-8
```



```
// Static method
static void showLibraryName() {
    System.out.println("Library Name (Static): " + libraryName);
}

// Final method (cannot be overridden)
final void showCategory() {
    System.out.println("Category (Final): " + category);
}

public static void main(String[] args) {
    System.out.println("----- Library Management System -----");

    // Create an object
    ModifiersApplication book = new ModifiersApplication("Java Programming", "Madhu", 101,
2023);

    // Access public method
    book.showTitle();

    // Access protected method
    book.displayDetails();

    // Access default variable
    System.out.println("Published Year (Default): " + book.publishedYear);

    // Access private data using private method
    book.showItemID();

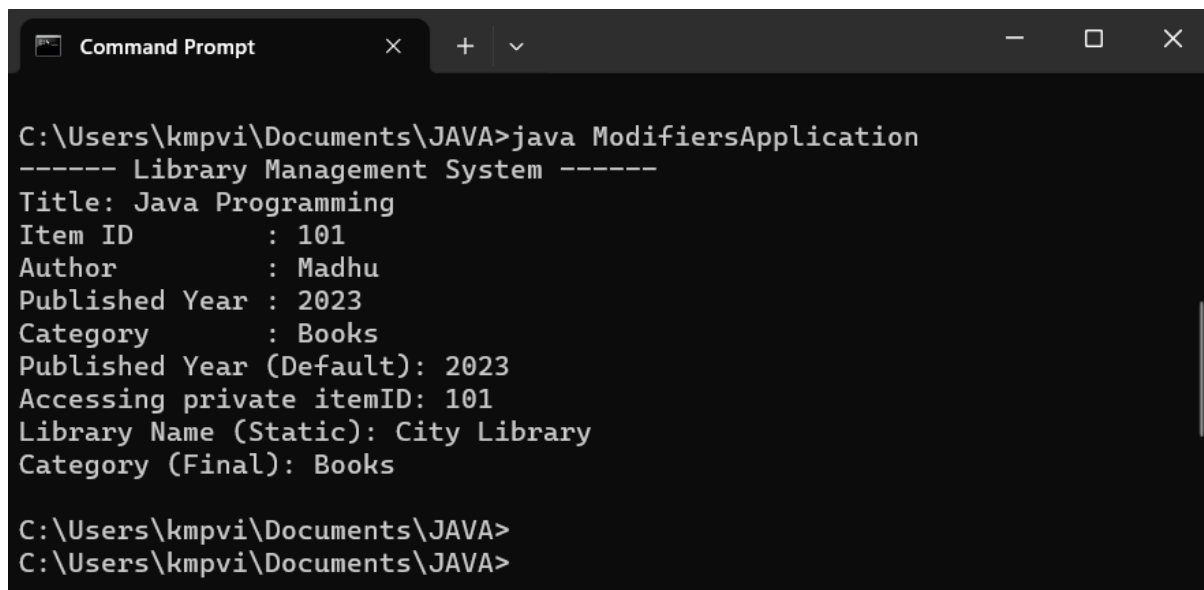
    // Access static method
    ModifiersApplication.showLibraryName();

    // Access final method
    book.showCategory();
}

}
```

Ln 73, Col 83 100% Windows (CRLF) UTF-8

Output:



```
C:\Users\kmpvi\Documents\JAVA>java ModifiersApplication
----- Library Management System -----
Title: Java Programming
Item ID      : 101
Author       : Madhu
Published Year : 2023
Category     : Books
Published Year (Default): 2023
Accessing private itemID: 101
Library Name (Static): City Library
Category (Final): Books

C:\Users\kmpvi\Documents\JAVA>
C:\Users\kmpvi\Documents\JAVA>
```


This program helps us understand how different **access modifiers** and **non-access modifiers** work in Java. It uses an **abstract class** (LibraryItem) with a protected variable and an abstract method, which is implemented in the child class ModifiersApplication.

The program shows:

- **Public members** like author and methods that can be accessed anywhere.
- **Private members** like itemID and showItemID() that are accessible only within the same class.
- **Protected members** like title, which are accessible within subclasses.
- **Default members** like publishedYear, accessible within the same package.
- **Static members** like libraryName, shared across all objects.
- **Final members** like category that cannot be modified or overridden.

By creating an object and calling these methods, the program demonstrates how access levels and modifiers control data visibility and method behavior in a real-world scenario like a library management system.

5) All Control Statements

Objective: To create a Java program that demonstrates the use of control statements such as if-else, switch-case, for, while, do-while, break, and continue through a real-world application.

Code :

```
1st lab practical works  BusBooking.java
File Edit View

import java.util.Scanner;

public class BusBooking {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        double totalAmount = 0.0;
        int seatAvailable = 10; // Total seats

        System.out.println(" Welcome to Madhu Travels");

        // Do-While for login
        int mainChoice;
        do {
            System.out.println("\n1. Book Ticket");
            System.out.println("2. Exit");
            System.out.print("Enter your choice: ");
            mainChoice = sc.nextInt();

            if (mainChoice == 1) {
                // While loop to allow booking while seats are available
                while (seatAvailable > 0) {
                    System.out.println("\nSeats Available: " + seatAvailable);
                    System.out.println("---- DESTINATIONS ----");
                    System.out.println("1. City A - Rs.200");
                    System.out.println("2. City B - Rs.300");
                    System.out.println("3. City C - Rs.400");
                    System.out.println("4. Cancel Booking");
                    System.out.print("Select destination (1-4): ");
                    int destChoice = sc.nextInt();
```

```
                // Switch-case for destinations
                switch (destChoice) {
                    case 1:
                        System.out.println(" Ticket to City A booked.");
                        totalAmount += 200;
                        seatAvailable--;
                        break;
                    case 2:
                        System.out.println(" Ticket to City B booked.");
                        totalAmount += 300;
                        seatAvailable--;
                        break;
                    case 3:
                        System.out.println(" Ticket to City C booked.");
                        totalAmount += 400;
                        seatAvailable--;
                        break;
                    case 4:
                        System.out.println(" Booking cancelled.");
                        continue; // Skip rest of this loop
                    default:
                        System.out.println("Invalid Choice! Try again.");
                        continue;
                }

                // If-Else for special discount
                if (totalAmount >= 1000) {
                    System.out.println(" You unlocked a ₹100 discount!");
                    totalAmount -= 100;
                }
            }
        } while (mainChoice != 2);
    }
}
```

```
1st lab practical works BusBooking.java
File Edit View

        totalAmount += 100;
    }

    System.out.print("Do you want to book another ticket? (yes=1 / no=0): ");
    int anotherTicket = sc.nextInt();
    if (anotherTicket == 0) {
        break; // Exit while loop
    }
}

if (seatAvailable == 0) {
    System.out.println("\n All seats booked! Try again later.");
}

// For loop to print booking confirmation
System.out.println("\n--- Printing Ticket ----");
for (int i = 1; i <= 3; i++) {
    System.out.println(" Ticket Copy " + i);
}

} else if (mainChoice == 2) {
    System.out.println("Thank you for visiting Madhu Travels. Goodbye!");
    break; // Exit do-while loop
} else {
    System.out.println("Invalid Choice! Please try again.");
}

} while (true);

// Final Amount
System.out.println("\n Payment ");
System.out.println("Total Amount: Rs" + totalAmount);
System.out.println("Enjoy your trip!");

sc.close();
}
}
```

Ln 85, Col 14 100% Windows (CRLF) UTF-8

Output:

```
Command Prompt
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kmpvi>cd C:\Users\kmpvi\Documents\JAVA

C:\Users\kmpvi\Documents\JAVA>javac BusBooking.java

C:\Users\kmpvi\Documents\JAVA>java BusBooking
Welcome to Madhu Travels

1. Book Ticket
2. Exit
Enter your choice: 1

Seats Available: 10
---- DESTINATIONS ----
1. City A - Rs.200
2. City B - Rs.300
3. City C - Rs.400
4. Cancel Booking
Select destination (1-4): 2
Ticket to City B booked.
Do you want to book another ticket? (yes=1 / no=0): 1

Seats Available: 9
---- DESTINATIONS ----
1. City A - Rs.200
2. City B - Rs.300
3. City C - Rs.400
4. Cancel Booking
Select destination (1-4): 1
Ticket to City A booked.
Do you want to book another ticket? (yes=1 / no=0): 1
```

```
Command Prompt
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kmpvi>cd C:\Users\kmpvi\Documents\JAVA

C:\Users\kmpvi\Documents\JAVA>javac BusBooking.java

C:\Users\kmpvi\Documents\JAVA>java BusBooking
Welcome to Madhu Travels

1. Book Ticket
2. Exit
Enter your choice: 1

Seats Available: 10
---- DESTINATIONS ----
1. City A - Rs.200
2. City B - Rs.300
3. City C - Rs.400
4. Cancel Booking
Select destination (1-4): 2
Ticket to City B booked.
Do you want to book another ticket? (yes=1 / no=0): 1

Seats Available: 9
---- DESTINATIONS ----
1. City A - Rs.200
2. City B - Rs.300
3. City C - Rs.400
4. Cancel Booking
Select destination (1-4): 1
Ticket to City A booked.
Do you want to book another ticket? (yes=1 / no=0): 1
```

```
Command Prompt

Seats Available: 8
---- DESTINATIONS ----
1. City A - Rs.200
2. City B - Rs.300
3. City C - Rs.400
4. Cancel Booking
Select destination (1-4): 4
Booking cancelled.

Seats Available: 8
---- DESTINATIONS ----
1. City A - Rs.200
2. City B - Rs.300
3. City C - Rs.400
4. Cancel Booking
Select destination (1-4): 3
Ticket to City C booked.
Do you want to book another ticket? (yes=1 / no=0): 0

---- Printing Ticket ----
Ticket Copy 1
Ticket Copy 2
Ticket Copy 3

1. Book Ticket
2. Exit
Enter your choice: 2
Thank you for visiting Madhu Travels. Goodbye!

Payment
Total Amount: Rs900.0
Enjoy your trip!

C:\Users\kmpvi\Documents\JAVA>
```

This program helps us understand how different control statements work in Java in a practical scenario. It uses the Scanner class to take user input and allows them to book bus tickets for various destinations.

The program demonstrates:

- **Do-while loop** for repeatedly showing the main menu (Book Ticket / Exit).
- **While loop** for allowing multiple bookings while seats are available.
- **Switch-case** for selecting destinations and calculating fares.
- **If-else** to check conditions like applying discounts when the total amount crosses ₹1000.
- **For loop** to print multiple ticket copies after booking.
- **Break and Continue statements** to control the program flow, such as canceling a booking or exiting loops early.

This program shows how control statements are used together in a real-world application like a Bus Ticket Booking System to make it interactive and user-friendly.