DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Engineering/Computer Science &

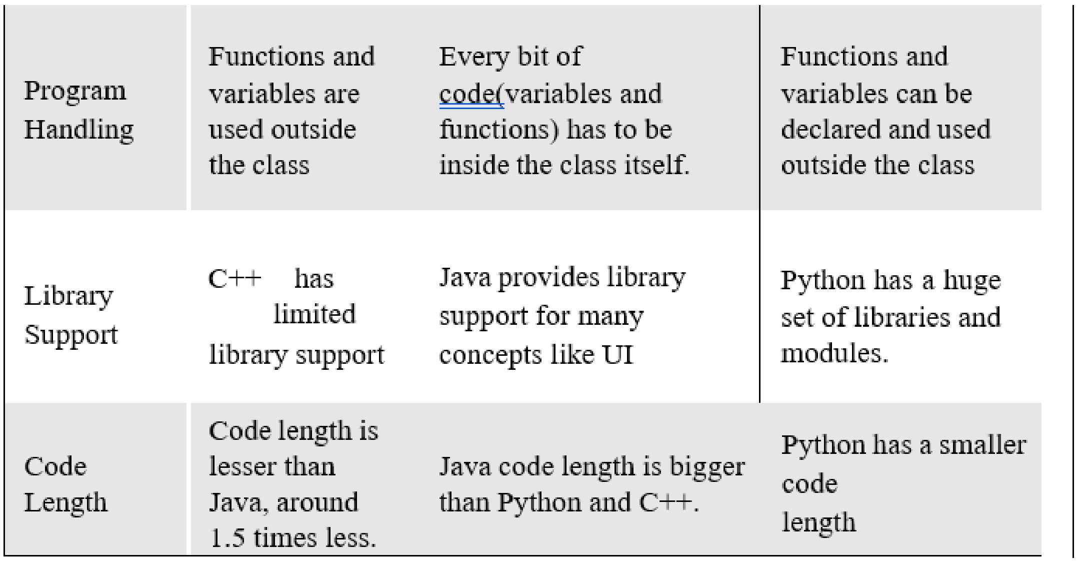
Engineering/Information Technology

Subject Name: JAVA PROGRAMMING
Semester: III
Subject Code: CSE201
Academic year: 2024-25

PART-I Data Types, Variables, String, Control Statements, Operators, Arrays

No.	Aim of the Practical
1.	**AIM :** Demonstration of installation steps of Java,Introduction to Object Oriented Concepts, comparison of Java with other object-oriented programming languages. Introduction to JDK, JRE, JVM, Javadoc, command line argument. Introduction to Eclipse or NetBeans IDE,or BlueJ and Console Programming. **Ans**. Object Oriented Concepts: Object-Oriented Programming (OOP) is a programming paradigm that uses objects and classes to design and develop applications. The key concepts of OOP are: • Class: A blueprint for creating objects. It defines a datatype by bundling data and methods that work on the data. • Object: An instance of a class. It represents a real-world entity. • Encapsulation: The wrapping of data (attributes) and methods (functions) into a single unit or class. It restricts direct access to some of the object's components, which is a means of preventing accidental interference and misuse of the data. • Inheritance: The mechanism by which one class can inherit the properties and behaviors of another class. It promotes code reusability. • Polymorphism: The ability of different classes to be treated as instances of the sameclass through inheritance. It allows one interface to be used for a general class of actions. Abstraction: The concept of hiding the complex implementation details and showing only the necessary features of an object. It simplifies the complexity of the system. Comparison of java with other object oriented programming languages:

TOPIC	C++	Java	Python
Compiled vs. Interpreted	Compiled Programming Language	Java is both Compiled and Interpreted.	Interpreted Programming Language
Platform Dependence	C++ is platform dependent	Java is platformindependent	Python is platformindependent
Operator Overloading	C++ supports operator overloading	Java does not support operator overloading	Python supports operator overloading
Inheritance	C++ provides both single and multiple inheritances	In Java, single inheritance is possible while multiple inheritances can be achieved using Interfaces	Python provides both single and multiple inheritances
Thread Support	C++ does not have built-in support for threads; It depends on Libraries	Java has built-in thread support	Python supports multithreading
Execution Time	C++ is very fast. It's, in fact, the first choice of competitive programmers	Java is much faster than Python in terms of speed of execution but slower than C++.	Due to the interpreter, Python is slow in terms of execution



Program Handling	Functions and variables are used outside the class	Every bit of code(variables and functions) has to be inside the class itself.	Functions and variables can be declared and used outside the class
Library Support	C++ has limited library support	Java provides library support for many concepts like UI	Python has a huge set of libraries and modules.
Code Length	Code length is lesser than Java, around 1.5 times less.	Java code length is bigger than Python and C++.	Python has a smaller code length

Introduction to JDK: The Java Development Kit (JDK) is a cross-platformed softwaredevelopment environment that offers a collection of tools and libraries necessary for developing Java-based software applications and applets. It is a core package used in Java, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment).

Beginners often get confused with JRE and JDK, if you are only interested in runningJava programs on your machine then you can easily do it using Java Runtime Environment.
However, if you would like to develop a Java-based software application then alongwith JRE you may need some additional necessary tools, which is called JDK.
The JDK has a private Java Virtual Machine (JVM) and a few other resources necessaryfor the development of a Java Application.
JDK contains:

- Java Runtime Environment (JRE),
- An interpreter/loader (Java),
- A compiler (javac),
- An archiver (jar) and many more.

The Java Runtime Environment in JDK is usually called Private Runtime because it is separated from the regular JRE and has extra content. The Private Runtime in JDK contains a JVM and all the class libraries present in the production environment, as wellas additional libraries useful to developers, e.g, internationalization libraries and the IDLlibraries.

Introduction to JRE: Java Runtime Environment (JRE) is an open-access software distribution that has a Java class library, specific tools, and a separate JVM. In Java, JREis one of the interrelated components in the Java Development Kit (JDK). It is the most common environment available on devices for running Java programs. Java source code
is compiled and converted to Java bytecode. If you want to run this bytecode on any platform, you need JRE. The JRE loads classes check memory access and get

systemresources. JRE acts as a software layer on top of the operating system.

Components of Java JRE
The components of JRE are mentioned below:

- Integration libraries include Java Database Connectivity (JDBC)
- Java Naming, Interface Definition Language (IDL)
- Directory Interface (JNDI)
- Remote Method Invocation Over Internet Inter-Orb Protocol (RMI-IIOP)
- Remote Method Invocation (RMI)
- Scripting

Java Virtual Machine (JVM**)** consists of Java HotSpot Client and Server VirtualMachine.

- User interface libraries include Swing, Java 2D, Abstract Window Toolkit (AWT), Accessibility, Image I/O, Print Service, Sound, drag, and Drop (DnD), and input methods.
- Lang and util base libraries, which include lang and util, zip, Collections, Concurrency Utilities, management, Java Archive (JAR), instrument, reflection, versioning, Preferences API, Ref Objects, Logging, and Regular Expressions.
- Other base libraries, including Java Management Extensions (JMX), Java Native Interface (JNI), Math, Networking, international support, input/output (I/O), Beans,
  Java Override Mechanism, Security, Serialization, extension mechanism, andJava for XML Processing (XML JAXP).
- Deployment technologies such as Java Web Start, deployment, and Java plug-in.

Introduction to JVM: JVM (Java Virtual Machine) is an abstract machine.
It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platformdependent).

It is:

- A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.
- An implementation Its implementation is known as JRE (Java Runtime Environment).
- Runtime Instance Whenever you write java command on the command prompt torun the java class, an instance of JVM is created.

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

Introduction to Javadoc: JavaDoc tool is a document generator tool in Java programming language for generating standard documentation in HTML format. It generates API documentation. It parses the declarations ad documentation in a set ofsource file describing classes, methods, constructors, and fields.

Before using JavaDoc tool, you must include JavaDoc comments /**…
....
*/
providing information about classes, methods, and constructors, etc. For creating a goodand understandable document API for any java file you must write better comments for every class, method, constructor.

The JavaDoc comments is different from the normal comments because of the extra asterisk at the beginning of the comment. It may contain the HTML tags as well.

By writing a number of comments, it does not affect the performance of the Javaprogram as all the comments are removed at compile time.

Introduction to command line argument: Java command-line argument is an argument i.e. passed at the time of running the Java program. In Java, the command linearguments passed from the console can be received in the Java program and they can beused as input. The users can pass the arguments during the execution bypassing the command-line arguments inside the main() method.

Working command-line arguments
We need to pass the arguments as space-separated values. We can pass both strings andprimitive data types(int, double, float, char, etc) as command-line arguments. These arguments convert into a string array and are provided to the main() function as a stringarray argument. When command-line arguments are supplied to JVM, JVM wraps theseand supplies them to args[]. It can be confirmed that they are wrapped up in an args array by checking the length of args using args.length.

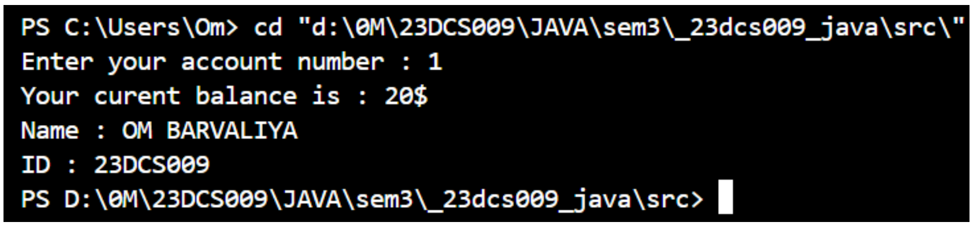
Internally, JVM wraps up these command-line arguments into the args[ ] array that we pass into the main() function. We can check these arguments using args.length method. JVM stores the first command-line argument at args[0], the second at args[1], the third at args[2], and so on.

2. **AIM :** Imagine you are developing a simple banking application where you need to display the current balance of a user account. For simplicity, let's say the current balance is $20. Write a java program to store this balance in a variable and then display it to the user.

PROGRAM CODE :

```
import java.util.Scanner;

public class Prac_02{
    public static void main(String[] args) {
        int balance=20,no;

        System.out.print("Enter your account number : ");
        Scanner s = new Scanner(System.in);
        no =s.nextInt();

        if(no==1)

    {
        System.out.print("Your curent balance is : "+ balance+"$");

    }
        else {System.out.print("Enter valid account number");}

        System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");
    }

}
```

OUTPUT:



```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\"
Enter your account number : 1
Your curent balance is : 20$
Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src>
```

OUTPUT: PRACTICAL-2

CONCLUSION:

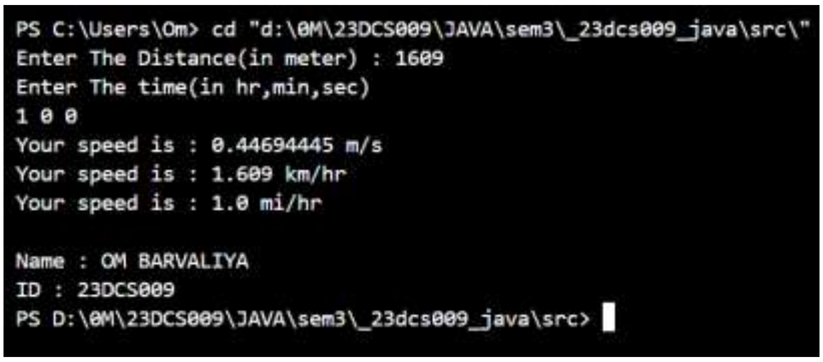
This code demonstrates a simple Java program that prompts the user for an account number, checks if it matches a predefined value (1), and then displays the current balance if the match is successful. If the account number does not match, it prompts the user to enter a valid account number. The program uses basic control flow with an if-else statement to determine the output based on the user's input. It also illustrates the use of the Scanner class for reading user input from the console. Additionally, the program prints the name and ID of the author or a placeholder individual, showcasing basic string concatenation and output in Java. This code serves as a fundamental example of conditional logic, user input

3. **AIM :** Write a program to take the user for a distance (in meters) and the time taken (as three numbers: hours, minutes, seconds), and display the speed, in meters per second, kilometers per hour and miles per hour (hint:1 mile = 1609 meters).

PROGRAM CODE :

```
import java.util.Scanner;

public class Prac_03{
public static void main(String[] args){
float distance;
int hr,min,sec;

System.out.print("Enter The Distance(in meter) : ");
Scanner s = new Scanner(System.in);
distance =s.nextFloat();
float km=distance/1000;
float miles=distance/1609;

System.out.println("Enter The time(in hr,min,sec)");
hr =s.nextInt();
min = s.nextInt();
sec = s.nextInt();
float timeInSec = (hr*3600)+(min*60)+(sec);
float timeInHr = hr+((float) min /60)+((float) sec /3600);

float speed1 = distance/timeInSec;
float speed2 =km/timeInHr;
float speed3 =miles/timeInHr;


System.out.println("Your speed is : "+speed1+" m/s");
System.out.println("Your speed is : "+speed2+" km/hr");
System.out.println("Your speed is : "+speed3+" mi/hr");
System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");
}}
```

OUTPUT:



```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\"
Enter The Distance(in meter) : 1609
Enter The time(in hr,min,sec)
1 0 0
Your speed is : 0.44694445 m/s
Your speed is : 1.609 km/hr
Your speed is : 1.0 mi/hr

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src>
```

OUTPUT: PRACTICAL-3

CONCLUSION:

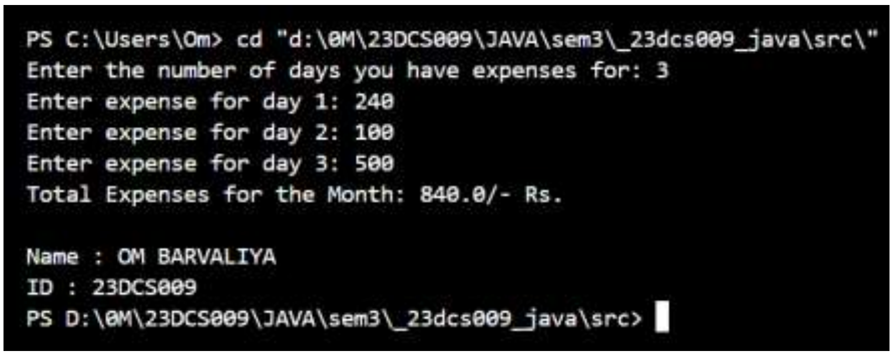
This program calculates and displays the speed of an object in meters per second, kilometers per hour, and miles per hour based on the user-provided distance (in meters) and time (in hours, minutes, and seconds). It utilizes the Scanner class for input, demonstrating basic Java input/output operations and arithmetic calculations. The conversion of distance to kilometers and miles, along with the conversion of time into seconds and hours, showcases practical applications of mathematical concepts in programming. The program outputs the calculated speeds in different units, providing a clear example of unit conversion and formatting output in Java. It concludes with the author's name and ID, personalizing the code output. This code snippet is a straightforward example of applying mathematical formulas and user input in Java to solve real-world problems.

4. **AIM :** Imagine you are developing a budget tracking application. You need to calculate the total expenses for the month. Users will input their daily expenses, and the program should compute the sum of these expenses. Write a Java program to calculate the sum of elements in an array representing daily expenses.

PROGRAM CODE :

```
import java.util.Scanner;

public class Prac_04 {

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

        System.out.print("Enter the number of days you have expenses for: ");
        int numberOfDays = scanner.nextInt();

        double[] dailyExpenses = new double[numberOfDays];

        for (int i = 0; i < numberOfDays; i++) {
            System.out.print("Enter expense for day " + (i + 1) + ": ");
            dailyExpenses[i] = scanner.nextDouble();
        }

        double totalExpenses = calculateTotalExpenses(dailyExpenses);

        System.out.println("Total Expenses for the Month: " + totalExpenses + "/-
Rs.");

        System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");
    }

  public static double calculateTotalExpenses(double[] expenses) {
    double total = 0;
    for (int i = 0; i < expenses.length; i++) {
        total += expenses[i];
    }
    return total;
  }
  }
```

OUTPUT:



```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\"
Enter the number of days you have expenses for: 3
Enter expense for day 1: 240
Enter expense for day 2: 100
Enter expense for day 3: 500
Total Expenses for the Month: 840.0/- Rs.

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src>
```

OUTPUT: PRACTICAL-4

CONCLUSION:

This program is designed to calculate and display the total monthly expenses based on daily inputs from the user. It starts by asking the user for the number of days they have expense data for, then collects expense amounts for each day using a loop. These amounts are stored in an array of doubles. The program calculates the total expenses by passing this array to the calculateTotalExpenses method, which iterates over the array to sum the expenses. Finally, it displays the total expenses to the user. This program demonstrates basic Java concepts such as arrays, loops, methods, and user input handling with the Scanner class. It concludes with displaying the author's name and ID, personalizing the output.

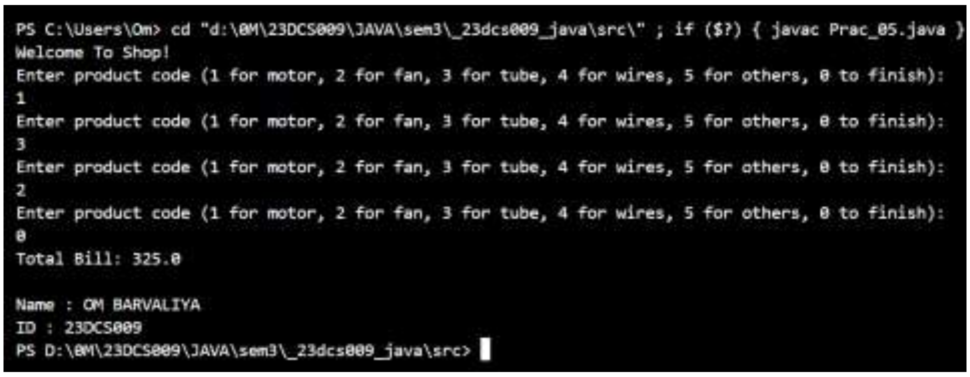
5.

AIM : An electric appliance shop assigns code 1 to motor,2 to fan,3 to tube and 4 for wires. All other items have code 5 or more. While selling the goods, a sales tax of 8% to motor,12% to fan,5% to tube light,7.5% to wires and 3% for all other items is charged. A list containing the product code and price in two different arrays. Write a java program using switch statement to prepare the bill.

PROGRAM CODE :

```
import java.util.Scanner;

public class Prac_05 {

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

        float[] price = {100, 100, 100, 100, 100};
        float totalPrice = 0;
        System.out.println("Welcome To Shop!");
        float tax = 0;
        int a;
        do {
            System.out.println("Enter product code (1 for motor, 2 for fan, 3 for tube, 4 
for wires, 5 for others, 0 to finish):");
            a = ip.nextInt();
            switch (a) {
                case 1:
                    tax = (price[0] * 8) / 100;
                    break;
                case 2:
                    tax = (price[1] * 12) / 100;
                    break;
                case 3:
                    tax = (price[2] * 5) / 100;
                    break;
                case 4:
                    tax = (price[3] * 7.5f) / 100;
                    break;
                case 5:
                    tax = (price[4] * 3) / 100;
                    break;
                default:
                    tax = 0;
                    break;
            }
            if (a > 0 && a <= 5) {
                totalPrice += price[a - 1] + tax;
            } else {
                totalPrice += 0;
            }
        } while (a != 0);

        System.out.println("Total Bill: " + totalPrice);
        System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");

    }
}
```

OUTPUT:



```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\" ; if ($?) { javac Prac_05.java }
Welcome To Shop!
Enter product code (1 for motor, 2 for fan, 3 for tube, 4 for wires, 5 for others, 0 to finish):
1
Enter product code (1 for motor, 2 for fan, 3 for tube, 4 for wires, 5 for others, 0 to finish):
3
Enter product code (1 for motor, 2 for fan, 3 for tube, 4 for wires, 5 for others, 0 to finish):
2
Enter product code (1 for motor, 2 for fan, 3 for tube, 4 for wires, 5 for others, 0 to finish):
0
Total Bill: 325.0

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src>
```

OUTPUT: PRACTICAL-5

CONCLUSION:

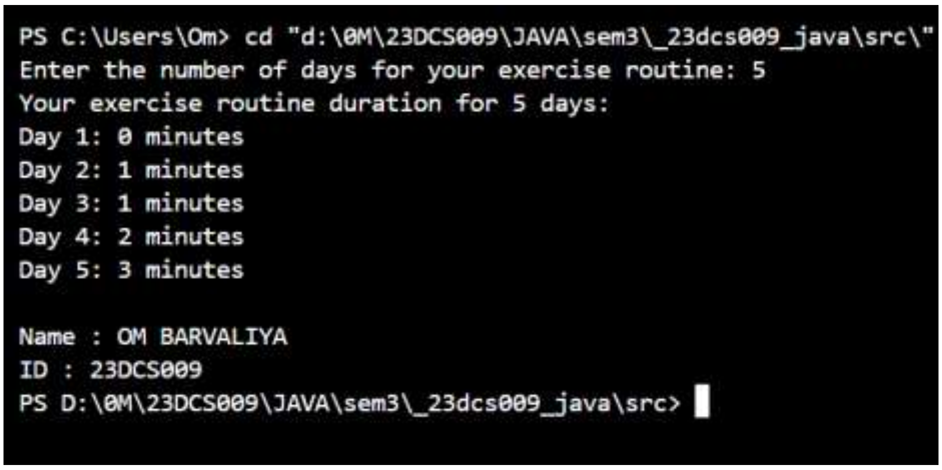
This program is a simple shopping cart calculator that computes the total bill including tax for a set of predefined products. It uses a do-while loop to continuously prompt the user for product codes until the user decides to finish by entering 0. Tax rates are applied differently based on the product selected, using a switch statement to determine the tax amount for each product. The total price, including tax, is accumulated and displayed at the end. This program demonstrates basic Java constructs such as arrays, loops, conditional statements, and user input handling with the Scanner class. It concludes by displaying the author's name and ID, adding a personal touch to the output.

6. **AIM :** Create a Java program that prompts the user to enter the number of days (n) for which they want to generate their exercise routine. The program should then calculate and display the first n terms of the Fibonacci series, representing the exercise duration for each day.

PROGRAM CODE :

```
import java.util.Scanner;

public class Prac_06 {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                System.out.print("Enter the number of days for your exercise routine:
");
                int n = scanner.nextInt();


                int firstTerm = 0, secondTerm = 1;
                System.out.println("Your exercise routine duration for " + n + "
days:");

                for (int i = 1; i <= n; ++i) {
                        System.out.println("Day " + i + ": " + firstTerm + " minutes");


                        int nextTerm = firstTerm + secondTerm;
                        firstTerm = secondTerm;
                        secondTerm = nextTerm;
                }
                System.out.print("\nName : OM BARVALIYA \nID : 23DCS009 ");
        }
}
```

OUTPUT:



```
PS C:\Users\Om> cd "d:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src\"
Enter the number of days for your exercise routine: 5
Your exercise routine duration for 5 days:
Day 1: 0 minutes
Day 2: 1 minutes
Day 3: 1 minutes
Day 4: 2 minutes
Day 5: 3 minutes

Name : OM BARVALIYA
ID : 23DCS009
PS D:\0M\23DCS009\JAVA\sem3\_23dcs009_java\src>
```

OUTPUT: PRACTICAL-6

CONCLUSION:

This program is designed to generate an exercise routine schedule based on the Fibonacci sequence, where the duration for each day is derived from this sequence. It prompts the user to enter the number of days they plan to follow the routine. Utilizing a simple for loop, it calculates the duration for each day, starting with 0 minutes on the first day and 1 minute on the second, then follows the Fibonacci pattern for subsequent days. The output is a daily exercise plan displaying the duration in minutes for the specified number of days. This program demonstrates the application of loops, basic arithmetic operations, and user input handling in Java. It concludes with displaying the author's name and ID, personalizing the output.