

# **Java Programs Collection**

**(with output)**

**Prepared by:**

**Sushma Kharat**

**College:**

**N.B.Navale Sinhgad College of  
Engineering , Solapur**

**University:**

**Punyashlok Ahilyadevi Holkar Solapur  
University**

**Date:**

**May 2025**

# Index

<b>Sr.No.</b>	<b>Program Title</b>	<b>Page No.</b>
1	calculate simple Interest	5
2	convert temperature from Fahrenheit to celsius	5
3	convert temperature from celsius to Fahrenheit	6
4	calculate electricity bill	6
5	ATM withdrawal	7,8
6	calculate an employee salary	9
7	Check for Even or Odd	11
8	Check Leap Year	11
9	Sum of Digits	12
10	Sum of Factors	13
11	Greatest of 3 Numbers	14
12	Check Triangle Type (Equilateral, Isosceles, Scalene)	15
13	Find Factors	17
14	Finding Factorial of a Number	17
15	Perfect Number Program	18
16	Armstrong Number Program	19
17	Neon Number Program	20
18	Disarium Number	21

<b>19</b>	<b>Munchausen Number</b>	<b>23</b>
<b>20</b>	<b>Reverse a Number</b>	<b>24</b>
<b>21</b>	<b>Days of the Week using Switch Statement</b>	<b>25</b>
<b>22</b>	<b>Program to print the letter N</b>	<b>26</b>
<b>23</b>	<b>Program to print the letter M</b>	<b>27</b>
<b>24</b>	<b>Program to print the letter K</b>	<b>28</b>
<b>25</b>	<b>Program to print the letter Z</b>	<b>29</b>
<b>26</b>	<b>Program to print the letter S</b>	<b>30</b>
<b>27</b>	<b>Program to print the letter W</b>	<b>31</b>
<b>28</b>	<b>Program to print the letter V</b>	<b>32</b>
<b>29</b>	<b>Bank Account</b>	<b>33</b>
<b>30</b>	<b>Bank Account Management System</b>	<b>39</b>
<b>31</b>	<b>Employee Salary Calculation</b>	<b>45</b>
<b>32</b>	<b>E-Commerce Order Processing</b>	<b>48</b>
<b>33</b>	<b>Create a Class SmartDevice with Overloaded Methods</b>	<b>53</b>
<b>34</b>	<b>Create a Base Class SmartDevice and Subclasses</b>	<b>54</b>
<b>35</b>	<b>Animal Abstract Class with Dog and Bird Subclasses</b>	<b>57</b>
<b>36</b>	<b>Shape Abstract Class with Circle and Rectangle Subclasses</b>	<b>59</b>
<b>37</b>	<b>Bank Account Interface with Saving Account and Current Account Classes</b>	<b>61</b>
<b>38</b>	<b>Vehicle interface with bike and car classes</b>	<b>64</b>
<b>39</b>	<b>Reverse a String</b>	<b>68</b>
<b>40</b>	<b>Check if a String is a Palindrome</b>	<b>68</b>
<b>41</b>	<b>Count Vowels and Consonants</b>	<b>69</b>
<b>42</b>	<b>Remove Duplicate Characters from a String</b>	<b>70</b>

<b>43</b>	<b>Check if Two Strings are Anagrams</b>	<b>71</b>
<b>44</b>	<b>Program to perform multiple risky operations</b>	<b>72</b>
<b>45</b>	<b>Program to Validate Voter Eligibility Using Exception Handling</b>	<b>74</b>
<b>46</b>	<b>Matrix Multiplication Program</b>	<b>75</b>
<b>47</b>	<b>Matrix Addition Program</b>	<b>78</b>
<b>48</b>	<b>program to display the names of all files and directories</b>	<b>81</b>
<b>49</b>	<b>program to display only file names</b>	<b>82</b>
<b>50</b>	<b>write text in file using filewriter</b>	<b>83,84</b>
<b>51</b>	<b>Reading data from file</b>	<b>85</b>
<b>52</b>	<b>Reading data from file using approach 2</b>	<b>88</b>
<b>53</b>	<b>Writing data in file using buffered writer</b>	<b>89</b>

### **Q1. Create a program that calculates simple Interest**

```
public class SimpleInterest {  
    public static void main(String[] args) {  
        float p,r,t,si;  
        p=13000;  
        r=12;  
        t=2;  
        si=(p*r*t)/100;  
        System.out.println("Simple Interest is:"+si);  
    }  
}
```

#### **Output:**

Simple Interest is:3120.0

### **Q2. Create a program that convert temperature from Fahrenheit to celsius**

```
public class Temperature {  
    public static void main(String[] args) {  
        float fahrenheit=(float) 75.2f;  
        float celsius=(fahrenheit-32)*5/9;  
        System.out.println(fahrenheit + "fahrenheit in celsius is:" + celsius);  
    }  
}
```

**Output:**

75.2 fahrenheit in celsius is: 23.999998

**Q3. Create a program that convert temperature from celsius to Fahrenheit**

```
public class Temperature {  
    public static void main(String[] args) {  
        float celsius=24;  
        float fahrenheit=32+(celsius*(9.0f/5));  
        System.out.println(celsius+"Celsius in Fahrenheit is: "+ fahrenheit);  
    }  
}
```

**Output:**

24.0 Celsius in Fahrenheit is: 75.2

**Q4. Write a program to calculate electricity bill based on following conditions:**

- i. Upto 100 units 5 Rs per unit**
- ii. 101 to 300 units 7 Rs per unit**
- iii. Above 300 units 10 Rs per unit**
- iv. If bill exceeds above 1000 add 10 Rs charge to total bill**
- v. Take the no of units in variable and display the final bill amount**

```
import java.util.Scanner;  
  
public class ElectricityBill {  
    public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);  
System.out.print("Enter the number of units consumed: ");  
int units = sc.nextInt();  
double bill = 0;  
if (units <= 100) {  
    bill = units * 5;  
} else if (units <= 300) {  
    bill = (100 * 5) + ((units - 100) * 7);  
} else {  
    bill = (100 * 5) + (200 * 7) + ((units - 300) * 10);  
} if (bill > 1000) {  
    bill += 10 }  
System.out.println("Your electricity bill is: Rs " + bill);  
}  
}
```

**Output:**

Enter the number of units consumed: 250

Your electricity bill is: Rs 1560.0

Enter the number of units consumed: 95

Your electricity bill is: Rs 475.0

Enter the number of units consumed: 426

Your electricity bill is: Rs 3170.0

**Q5. Write a program that simulates an ATM withdrawal take the withdrawal amount and available balance as input and check the following conditions:**

- i. Withdrawal amount should be in multiple of 100**
- ii. Withdrawal amount should not exceed available balance**
- iii. If transaction is successful, display remaining balance otherwise print an appropriate error.**

```
import java.util.Scanner;

public class ATMWithdrawl {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your available balance: Rs ");
        double balance = sc.nextDouble();

        System.out.print("Enter the amount to withdraw: Rs ");
        double withdrawalAmount = sc.nextDouble();

        if (withdrawalAmount % 100 != 0) {

            System.out.println("Error: Withdrawal amount should be in multiples of 100.");

        } else if (withdrawalAmount > balance) {

            System.out.println("Error: Insufficient balance.");

        } else {

            balance -= withdrawalAmount;

            System.out.println("Transaction successful.");

            System.out.println("Remaining balance: Rs " + balance);

        }

    }

}
```



}

**Output:**

Enter your available balance: Rs 2450

Enter the amount to withdraw: Rs 400

Transaction successful.

Remaining balance: Rs 2050.0

Enter your available balance: Rs 23000

Enter the amount to withdraw: Rs 31000

Error: Insufficient balance.

Enter your available balance: Rs 12345

Enter the amount to withdraw: Rs 2357

Error: Withdrawal amount should be in multiples of 100.

**Q6. Write a program to calculate an employee salary after applying taxes based on following conditions:**

**i. If salary greater than 1000 add 5% tax**

**ii. If salary between 10001 and 50000 add 10% tax**

**iii. If above 50000 add 20% tax**

**iv. Take employee basic salary input and compute the tax and print the salary**

```
import java.util.Scanner;
```

```
public class EmployeeSalary {
```

```
    public static void main(String[] args) {
```

```

Scanner sc = new Scanner(System.in);
System.out.print("Enter employee basic salary: Rs ");
double salary = sc.nextDouble();
double tax = 0;
if (salary > 50000) {
    tax = salary * 0.20;
} else if (salary >= 10001 && salary <= 50000) {
    tax = salary * 0.10;
} else if (salary > 1000) {
    tax = salary * 0.05;
}
double finalSalary = salary - tax;
System.out.println("Tax applied: Rs " + tax);
System.out.println("Final salary after tax: Rs " + finalSalary);
}
}

```

### **Output:**

Enter employee basic salary: Rs 34000

Tax applied: Rs 3400.0

Final salary after tax: Rs 30600.0

Enter employee basic salary: Rs 75390

Tax applied: Rs 15078.0

Final salary after tax: Rs 60312.0

Enter employee basic salary: Rs 5723

Tax applied: Rs 286.15000000000003

Final salary after tax: Rs 5436.85

### **Q7.Check for Even or Odd:**

```
import java.util.Scanner;

public class EvenOddCheck {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int num = sc.nextInt();

        if (num % 2 == 0) {

            System.out.println(num + " is Even.");

        } else {

            System.out.println(num + " is Odd.");

        }

    }

}
```

### **Output:**

Enter a number: 26

26 is Even.

Enter a number: 433

433 is Odd.

### **Q8. Check Leap Year:**

```
import java.util.Scanner;

public class LeapYearCheck {

    public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter a year: ");
int year = sc.nextInt();
if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
    System.out.println(year + " is a Leap Year.");
} else {
    System.out.println(year + " is not a Leap Year.");
}
}
```

**Output:**

Enter a year: 2025  
2025 is not a Leap Year.

Enter a year: 2040  
2040 is a Leap Year.

**Q9. Sum of Digits:**

```
import java.util.Scanner;
public class SumOfDigits {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int no = sc.nextInt();
        int sum = 0;
```

```

while (no > 0) {
    int rem = no % 10;
    sum += rem;
    no /= 10;
}
System.out.println(sum + " is the sum of digits of the given number.");
}
}

```

### **Output:**

Enter a number: 1234

10 is the sum of digits of the given number.

Enter a number: 26625

21 is the sum of digits of the given number.

### **Q10. Sum of Factors:**

```

import java.util.Scanner;

public class SumOfFactors {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int no = sc.nextInt();
        int sum = 0;
        for (int i = 1; i < no; i++) {
            if (no % i == 0) {

```

```
        sum += i;
    }
}

System.out.println(sum + " is the sum of factors of " + no + ".");
}
}
```

**Output:**

Enter a number: 26

16 is the sum of factors of 26.

Enter a number: 64

63 is the sum of factors of 64.

**Q11. Greatest of 3 Numbers:**

```
import java.util.Scanner;

public class GreatestOfThree {

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

        System.out.print("Enter the first number: ");
        int num1 = sc.nextInt();

        System.out.print("Enter the second number: ");
        int num2 = sc.nextInt();

        System.out.print("Enter the third number: ");
        int num3 = sc.nextInt();

        int greatest;
```

```

    if (num1 >= num2 && num1 >= num3) {
        greatest = num1;
    } else if (num2 >= num1 && num2 >= num3) {
        greatest = num2;
    } else {
        greatest = num3;
    }
    System.out.println("The greatest number is: " + greatest);
}
}

```

### **Output:**

```

Enter the first number: 26
Enter the second number: 31
Enter the third number: 67215
The greatest number is: 67215

```

### **Q12.Check Triangle Type (Equilateral, Isosceles, Scalene):**

```

import java.util.Scanner;

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

        System.out.print("Enter the first side of the triangle: ");
        int side1 = sc.nextInt();

        System.out.print("Enter the second side of the triangle: ");
        int side2 = sc.nextInt();
    }
}

```

```
System.out.print("Enter the third side of the triangle: ");
int side3 = sc.nextInt();
if (side1 == side2 && side2 == side3) {
    System.out.println("The triangle is Equilateral.");
} else if (side1 == side2 || side2 == side3 || side1 == side3) {
    System.out.println("The triangle is Isosceles.");
} else {
    System.out.println("The triangle is Scalene.");
}
}
}
```

**Output:**Enter the first side of the triangle: 15

Enter the second side of the triangle: 15

Enter the third side of the triangle: 15

The triangle is Equilateral.

Enter the first side of the triangle: 23

Enter the second side of the triangle: 26

Enter the third side of the triangle: 30

The triangle is Scalene.

Enter the first side of the triangle: 6

Enter the second side of the triangle: 6

Enter the third side of the triangle: 13

The triangle is Isosceles.



### **Q13. Print Factors of Inputted Number:**

```
import java.util.Scanner;

public class PrintFactors {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int no = sc.nextInt();

        System.out.println("The factors of " + no + " are:");

        for (int i = 1; i <= no; i++) {

            if (no % i == 0) {

                System.out.print(i + " ");

            }

        }

        System.out.println();

    }

}
```

#### **Output:**

Enter a number: 48

The factors of 48 are:

1 2 3 4 6 8 12 16 24 48

### **Q14. Finding Factorial of a Number:**

```
import java.util.Scanner;

public class Factorial {

    public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter a number to find its factorial: ");
int num = sc.nextInt();
long factorial = 1;
for (int i = 1; i <= num; i++) {
    factorial *= i;
}
System.out.println("The factorial of " + num + " is: " + factorial);
}
}
```

**Output:**

Enter a number to find its factorial: 7

The factorial of 7 is: 5040

Enter a number to find its factorial: 4

The factorial of 4 is: 24

**Q15. Perfect Number Program**

```
import java.util.Scanner;
public class PerfectNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number to check if it's a Perfect number: ");
        int num = sc.nextInt();
        int sum = 0;
```

```

    for (int i = 1; i < num; i++) {
        if (num % i == 0) {
            sum += i;
        }
    }

    if (sum == num) {
        System.out.println(num + " is a Perfect number.");
    } else {
        System.out.println(num + " is not a Perfect number.");
    }
}
}

```

**Output:** Enter a number to check if it's a Perfect number: 28

28 is a Perfect number.

Enter a number to check if it's a Perfect number: 67

67 is not a Perfect number.

### **Q16. Armstrong Number Program**

```

import java.util.Scanner;

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

        System.out.print("Enter a number to check if it's an Armstrong number: ");

        int num = sc.nextInt();

        int temp = num, sum = 0;

        while (num > 0) {

```

```

        int rem = num % 10;

        sum += rem * rem * rem;

        num /= 10;

    }if (temp == sum) {

        System.out.println(temp + " is an Armstrong number.");

    } else {

        System.out.println(temp + " is not an Armstrong number."); }

    } }

```

### **Output:**

Enter a number to check if it's an Armstrong number: 153

153 is an Armstrong number.

Enter a number to check if it's an Armstrong number: 371

371 is an Armstrong number.

Enter a number to check if it's an Armstrong number: 135

135 is not an Armstrong number.

### **Q17. Neon Number Program:**

```

import java.util.Scanner;

public class NeonNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number to check if it's a Neon number: ");

        int num = sc.nextInt();

        int sum = 0;

        int sqr = num * num;

```

```

while (sqr > 0) {
    sum += sqr % 10;
    sqr /= 10;
}if (sum == num) {
    System.out.println(num + " is a Neon number.");
} else {
    System.out.println(num + " is not a Neon number.");
}
}
}

```

### **Output:**

Enter a number to check if it's a Neon number: 9

9 is a Neon number.

Enter a number to check if it's a Neon number: 25

25 is not a Neon number.

### **Q18. Disarium Number Program:**

```

import java.util.Scanner;

public class DisariumNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number to check if it's a Disarium number: ");
        int num = sc.nextInt();
        int temp = num, count = 0, sum = 0;
        while (temp > 0) {

```

```
        count++;
        temp /= 10;
    }temp = num;
    for (int i = count; i > 0; i--) {
        int digit = temp % 10;
        sum += Math.pow(digit, i);
        temp /= 10;
    }
    if (sum == num)
    {
        System.out.println(num + " is a Disarium number.");
    }
    else {
        System.out.println(num + " is not a Disarium number.");
    }
}
```

**Output:**

Enter a number to check if it's a Disarium number: 518

518 is a Disarium number.

Enter a number to check if it's a Disarium number: 598

598 is a Disarium number.

Enter a number to check if it's a Disarium number: 158

158 is not a Disarium number.

### **Q19.Munchausen Number Program:**

```
import java.util.Scanner;

public class MunchausenNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number to check a Munchausen number: ");

        int num = sc.nextInt();

        int sum = 0, temp = num;

        for (char digit : String.valueOf(num).toCharArray()) {

            int d = Character.getNumericValue(digit);

            sum += Math.pow(d, d);

        }if (sum == temp) {

            System.out.println(num + " is a Munchausen number.");

        } else {

            System.out.println(num + " is not a Munchausen number.");

        }

    }

}
```

#### **Output:**

Enter a number to check a Munchausen number: 3435

3435 is a Munchausen number.

Enter a number to check a Munchausen number: 315

315 is not a Munchausen number.

### **Q20. Reverse a Number Program:**

```
import java.util.Scanner;

public class ReverseNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number to reverse it: ");

        int num = sc.nextInt();

        int reversed = 0;

        while (num != 0) {

            int rem = num % 10;

            reversed = reversed * 10 + rem;

            num /= 10;

        }

        System.out.println("Reversed number: " + reversed);

    }

}
```

#### **Output:**

Enter a number to reverse it: 1234

Reversed number: 4321

Enter a number to reverse it: 2662005

Reversed number: 5002662



### **Q21. Days of the Week using Switch Statement:**

```
import java.util.Scanner;

public class DaysOfWeek {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number between 1 and 7 to get the day of the
week: ");

        int dayNumber = sc.nextInt();

        switch (dayNumber) {

            case 1:

                System.out.println("Monday");

                break;

            case 2:

                System.out.println("Tuesday");

                break;

            case 3:

                System.out.println("Wednesday");

                break;

            case 4:

                System.out.println("Thursday");

                break;

            case 5:

                System.out.println("Friday");

                break;

            case 6:

                System.out.println("Saturday");
```

```

        break;
    case 7:
        System.out.println("Sunday");
        break;
    default:
        System.out.println("Invalid input! Please enter a number between 1
and 7.");
        break;
    }
}

```

### **Output:**

Enter a number between 1 and 7 to get the day of the week: 7

Sunday

Enter a number between 1 and 7 to get the day of the week: 3

Wednesday

### **Q22. Program to print the letter N:**

```

public class Letterpattern {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            for (int j = 1; j <= 5; j++) {
                if (j == 1 || j == 5 || i == j) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
        }
    }
}

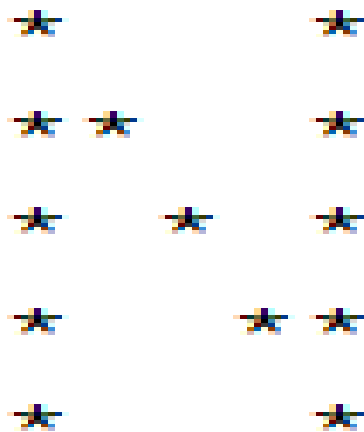
```

```

    }
}
System.out.println();
}
}
}

```

### **Output:**



### **Q23. Program to print the letter M:**

```

public class M {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            for (int j = 1; j <= 5; j++) {
                if (j == 1 || j == 5 || i == j & j <= 3 || j == 6 - i & j > 3) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
        }
    }
}

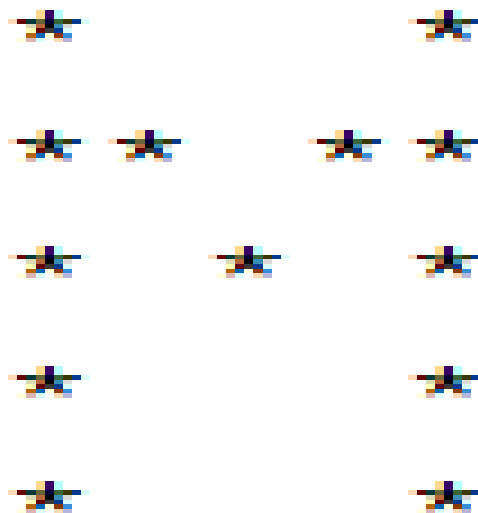
```

```

    }
}
System.out.println(" ");
}
}
}
}

```

### **Output:**



### **Q24. Program to print the letter K:**

```

public class Kpattern {
    public static void main(String[] args) {
        for (int i = 1; i <= 7; i++) {
            for (int j = 1; j <= 7; j++) {
                if (j == 1 || (j==6-i) || (i==2+j)) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
        }
    }
}

```

```

    }
}
System.out.println("");
}
}
}

```

### **Output:**



### **Q25. Program to print the letter Z:**

```

public class Z {
    public static void main(String[] args) {
        for (int i = 1; i <= 7; i++) {
            for (int j = 1; j <= 7; j++) {
                if (i == 1 || i == 7 || i + j == 8) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}

```

```

    }
}
System.out.println();
}
}
}

```

### **Output:**

```

      ★ ★ ★ ★ ★ ★ ★
        ★
          ★
            ★
              ★
                ★
      ★ ★ ★ ★ ★ ★ ★

```

### **Q26. Program to print the letter S:**

```

public class S {
    public static void main(String[] args) {
        for (int i = 1; i <= 7; i++) {
            for (int j = 1; j <= 7; j++) {
                if ((i == 1) || (i == 7) || (i == 4) || (i < 4 && j == 1) || (i > 4 && j == 7))
                {
                    System.out.print("*");
                } else {

```

```

        System.out.print(" ");
    }
}
System.out.println();
}
}
}

```

### **Output:**

```

    ★ ★ ★ ★ ★ ★ ★
    ★
    ★
    ★ ★ ★ ★ ★ ★ ★
                                     ★
                                     ★
    ★ ★ ★ ★ ★ ★ ★

```

### **Q27. Program to print the letter W:**

```

public class W {
    public static void main(String[] args) {
        for (int i = 1; i <= 7; i++) {
            for (int j = 1; j <= 7; j++) {
                if (j == 1 || j == 7 || i == j && i >= 4 || i + j == 8 && i >= 4) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
        }
    }
}

```

```

    }
}
System.out.println();
}
}
}

```

### **Output:**

```

      ★          ★
      ★          ★
      ★          ★
      ★      ★  ★  ★
      ★    ★  ★  ★  ★
      ★  ★      ★  ★
      ★          ★

```

### **Q28. Program to print the letter :V**

```

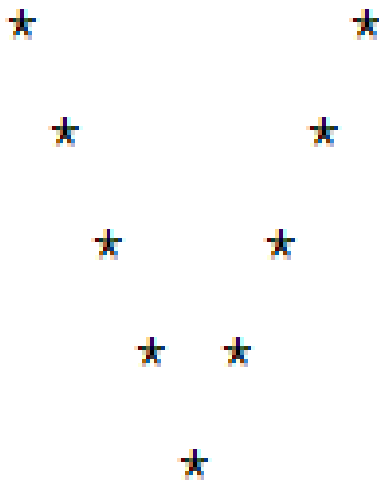
public class V {
    public static void main(String[] args) {
        int n = 5;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < i; j++) System.out.print(" ");
            System.out.print("*");
            for (int j = 0; j < 2 * (n - i - 1) - 1; j++) System.out.print(" ");
            if (i != n - 1) System.out.println("*");
        }
    }
}

```



```
        else System.out.println();  
    }  
}  
}
```

**Output:**



A 5x5 grid of stars forming a diamond shape. The stars are arranged in a pattern where the number of stars per row increases from 1 to 5 and then decreases back to 1. The stars are colored with a gradient from blue to yellow.

**Q29.Bank Account:**

```
import java.util.Scanner;  
public class BankAccount {  
  
    // Instance variables  
    private String accountHolderName;  
    private String accountNo;  
    private double balance;
```

```

// Static variable
static String bankName;

// Constructor to initialize the bank account
public BankAccount(String accountHolderName, String accountNo, double
balance) {
    this.accountHolderName = accountHolderName;
    this.accountNo = accountNo;
    this.balance = balance;
}

// Instance method: Deposit money into the account
public void deposit(double amount) {
    if (amount > 0)
    {
        balance += amount;
        System.out.println(amount + " deposited successfully.");
    }
Else
{
    System.out.println("Deposit amount must be positive.");
}
}

// Instance method: Withdraw money from the account
public void withdraw(double amount)

```

```

{
    if (amount > 0 && amount <= balance)
    {
        balance -= amount;

        System.out.println(amount + " withdrawn successfully.");
    }
else
{
    System.out.println("Insufficient balance or invalid withdrawal
amount.");
}
}

// Instance method: Display account information
public void displayAccountInfo() {
    System.out.println("\nAccount Holder Name: " + accountHolderName);
    System.out.println("Account Number: " + accountNo);
    System.out.println("Balance: " + balance);
    System.out.println("Bank Name: " + bankName);
}

// Static method: Change the bank name for all accounts
public static void changeBankName(String newBankName) {
    bankName = newBankName;
    System.out.println("Bank name changed to: " + bankName);
}

public static void main(String[] args)

```

```

{
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the bank name: ");

    String initialBankName = sc.nextLine();
    BankAccount.changeBankName(initialBankName);

    System.out.print("\nEnter account holder name: ");
    String accountHolderName1 = sc.nextLine();

    System.out.print("Enter account number: ");
    String accountNo1 = sc.nextLine();

    System.out.print("Enter initial balance: ");
    double balance1 = sc.nextDouble();

    sc.nextLine();

    BankAccount account1 = new BankAccount(accountHolderName1,
    accountNo1, balance1);
    account1.displayAccountInfo();

    // Deposit money
    System.out.print("\nEnter deposit amount for " + accountHolderName1 +
    ": ");
    double depositAmount1 = sc.nextDouble();
    account1.deposit(depositAmount1);

```

```

        account1.displayAccountInfo();

// Withdraw money

        System.out.print("\nEnter withdrawal amount for " +
accountHolderName1 + ": ");

        double withdrawAmount1 = sc.nextDouble();

        account1.withdraw(withdrawAmount1);

        account1.displayAccountInfo();


// Change bank name

        sc.nextLine();

        System.out.print("\nEnter new bank name: ");

        String newBankName = sc.nextLine();

        BankAccount.changeBankName(newBankName);


// Take input for the second account to see the change in bank name

        System.out.print("\nEnter account holder name for second account: ");

        String accountHolderName2 = sc.nextLine();

        System.out.print("Enter account number for second account: ");

        String accountNo2 = sc.nextLine();

        System.out.print("Enter initial balance for second account: ");

        double balance2 = sc.nextDouble();

        BankAccount account2 = new BankAccount(accountHolderName2,
accountNo2, balance2);

        account2.displayAccountInfo();

    }

}

```

**Output:**

Enter the bank name: SBI

Bank name changed to: SBI

Enter account holder name: sushma

Enter account number: 12345656366

Enter initial balance: 35845

Account Holder Name: sushma

Account Number: 12345656366

Balance: 35845.0

Bank Name: SBI

Enter deposit amount for sushma: 2500

2500.0 deposited successfully.

Account Holder Name: sushma

Account Number: 12345656366

Balance: 38345.0

Bank Name: SBI

Enter withdrawal amount for sushma: 30000

30000.0 withdrawn successfully.

Account Holder Name: sushma

Account Number: 12345656366

Balance: 8345.0

Bank Name: SBI

Enter new bank name: BOI

Bank name changed to: BOI

Enter account holder name for second account: sushma2

Enter account number for second account: 2345657777777777777777

Enter initial balance for second account: 00

Account Holder Name: sushma2

Account Number: 2345657777777777777777

Balance: 0.0

Bank Name: BOI

**Q30. develop a Java program for a Bank Account Management System that demonstrates constructor overloading and method overloading:**

```
import java.util.Scanner;

public class BankAccount {

    // Instance variables
    private String accountNo;
    private double balance;

    // Task 1: Constructor Overloading
    // Default constructor
    public BankAccount() {
        this.accountNo = "000000";
        this.balance = 0.0;
    }

    // Parameterized constructor (1 parameter)
```

```
public BankAccount(String accountNo) {  
    this.accountNo = accountNo;  
    this.balance = 0.0;  
}
```

```
// Parameterized constructor (2 parameters)
```

```
public BankAccount(String accountNo, double initialBalance) {  
    this.accountNo = accountNo;  
    this.balance = initialBalance;  
}
```

```
// Task 2: Method Overloading (Deposit Methods)
```

```
// Method to deposit an integer amount
```

```
public void deposit(int amount) {  
    if (amount > 0) {  
        balance += amount;  
        System.out.println(amount + " deposited successfully.");  
    } else {  
        System.out.println("Deposit amount must be positive.");  
    }  
}
```

```
// Method to deposit a double amount
```

```
public void deposit(double amount) {  
    if (amount > 0) {
```



```

        balance += amount;

        System.out.println(amount + " deposited successfully.");
    } else {
        System.out.println("Deposit amount must be positive.");
    }
}

// Method to deposit an amount from a string (converts string to double)
public void deposit(String amount) {
    try {
        double depositAmount = Double.parseDouble(amount);
        if (depositAmount > 0) {
            balance += depositAmount;
            System.out.println(depositAmount + " deposited successfully.");
        } else {
            System.out.println("Deposit amount must be positive.");
        }
    } catch (NumberFormatException e) {
        System.out.println("Invalid amount. Please enter a valid number.");
    }
}

// Method to display account details
public void displayAccountInfo() {
    System.out.println("\nAccount Number: " + accountNo);
    System.out.println("Balance: " + balance);
}

```

```

}

public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

    // Creating BankAccount instances using different constructors
    BankAccount account1 = new BankAccount();
    BankAccount account2 = new BankAccount("123456789");
    BankAccount account3 = new BankAccount("987654321", 500.0);

    // Display account info before deposits
    System.out.println("Account 1 (Default Constructor):");
    account1.displayAccountInfo();

    System.out.println("Account 2 (1 Parameter Constructor):");
    account2.displayAccountInfo();

    System.out.println("Account 3 (2 Parameters Constructor):");
    account3.displayAccountInfo();

    // Testing deposit method (overloading)
    System.out.print("\nEnter an integer amount to deposit into Account 1: ");
    int depositAmountInt = sc.nextInt();
    account1.deposit(depositAmountInt);

    System.out.print("\nEnter a double amount to deposit into Account 2: ");
    double depositAmountDouble = sc.nextDouble();

```

```
account2.deposit(depositAmountDouble);

// Using String as input for deposit method
System.out.print("\nEnter a deposit amount (as string) for Account 3: ");
sc.nextLine(); // consume newline
String depositAmountString = sc.nextLine();
account3.deposit(depositAmountString);

// Display updated account info after deposits
System.out.println("\nUpdated Account 1 Info:");
account1.displayAccountInfo();

System.out.println("Updated Account 2 Info:");
account2.displayAccountInfo();

System.out.println("Updated Account 3 Info:");
account3.displayAccountInfo();
}
}
```

**Output:**

Account 1 (Default Constructor):

Account Number: 000000

Balance: 0.0

Account 2 (1 Parameter Constructor):

Account Number: 123456789

Balance: 0.0

Account 3 (2 Parameters Constructor):

Account Number: 987654321

Balance: 500.0

Enter an integer amount to deposit into Account 1: 345

345 deposited successfully.

Enter a double amount to deposit into Account 2: 456

456.0 deposited successfully.

Enter a deposit amount (as string) for Account 3: 6556

6556.0 deposited successfully.

Updated Account 1 Info:

Account Number: 000000

Balance: 345.0

Updated Account 2 Info:

Account Number: 123456789

Balance: 456.0

Updated Account 3 Info:

Account Number: 987654321

Balance: 7056.0

### **Q31.Employee Salary Calculation:**

```
public class Employee {  
    // Instance variables  
    private String name;  
    private double salary;  
    // Constructor Overloading  
    // Default constructor  
    public Employee() {  
        this.name = "Unknown";  
        this.salary = 0.0;  
    }  
    // Constructor with name parameter  
    public Employee(String name) {  
        this.name = name;  
        this.salary = 25000.0; // Default salary for this constructor  
    }  
    // Constructor with name and salary parameters  
    public Employee(String name, double salary) {  
        this.name = name;  
        this.salary = salary;  
    }  
    // Method Overloading  
    // calculateBonus() - returns 10% of salary as bonus  
    public double calculateBonus() {  
        return 0.10 * salary;  
    }  
}
```

```

    // calculateBonus(double percentage) - returns bonus based on the given
percentage

    public double calculateBonus(double percentage) {
        return (percentage / 100) * salary;
    }

    // calculateBonus(double percentage, double additionalBonus) - returns
total bonus including an additional bonus

    public double calculateBonus(double percentage, double additionalBonus) {
        return (percentage / 100) * salary + additionalBonus;
    }

// Method to display employee details

    public void displayEmployeeInfo() {
        System.out.println("Employee Name: " + name);
        System.out.println("Employee Salary: " + salary);
    }

public static void main(String[] args) {

    // Creating employee instances using different constructors

    Employee emp1 = new Employee(); // Using default constructor

    Employee emp2 = new Employee("John Doe"); // Using constructor with
name parameter

    Employee emp3 = new Employee("Jane Smith", 50000.0);

// Using constructor with name and salary parameters

    // Display employee information

    System.out.println("Employee 1 (Default Constructor):");

    emp1.displayEmployeeInfo();

    System.out.println("Bonus: " + emp1.calculateBonus());

```

```
System.out.println("\nEmployee 2 (Constructor with Name):");
emp2.displayEmployeeInfo();
System.out.println("Bonus: " + emp2.calculateBonus());
System.out.println("\nEmployee 3 (Constructor with Name and Salary):");
emp3.displayEmployeeInfo();
System.out.println("Bonus (10%): " + emp3.calculateBonus());
System.out.println("Bonus (15%): " + emp3.calculateBonus(15));
System.out.println("Bonus (15%, Additional 500): " +
emp3.calculateBonus(15, 500));
}
}
```

### **Output:**

Employee 1 (Default Constructor):

Employee Name: Unknown

Employee Salary: 0.0

Bonus: 0.0

Employee 2 (Constructor with Name):

Employee Name: John Doe

Employee Salary: 25000.0

Bonus: 2500.0

Employee 3 (Constructor with Name and Salary):

Employee Name: Jane Smith

Employee Salary: 50000.0

Bonus (10%): 5000.0

Bonus (15%): 7500.0

Bonus (15%, Additional 500): 8000.0

### **Q32.E-Commerce Order Processing**

```
public class Order {  
    // Instance variables  
    private int orderId;  
    private double amount;  
  
    // Constructor Overloading  
  
    // Default constructor  
    public Order() {  
        this.orderId = 0;  
        this.amount = 0.0;  
    }  
  
    // Constructor with orderId parameter  
    public Order(int orderId) {  
        this.orderId = orderId;  
        this.amount = 1000.0; // Default amount for this constructor  
    }  
  
    // Constructor with orderId and amount parameters  
    public Order(int orderId, double amount) {  
        this.orderId = orderId;  
        this.amount = amount;  
    }  
}
```



```
// Method Overloading for applyDiscount

// applyDiscount() - Applies 5% discount by default
public void applyDiscount() {
    amount -= 0.05 * amount;
    System.out.println("5% discount applied. Updated amount: " + amount);
}

// applyDiscount(int discount) - Applies the given discount percentage
public void applyDiscount(int discount) {
    amount -= (discount / 100.0) * amount;
    System.out.println(discount + "% discount applied. Updated amount: " +
amount);
}

// applyDiscount(int discount, boolean isFirstOrder) - If it's the first order,
adds 5% extra discount
public void applyDiscount(int discount, boolean isFirstOrder) {
    if (isFirstOrder) {
        discount += 5; // Add 5% extra discount if it's the first order
        System.out.println("First order! Adding 5% extra discount.");
    }
    amount -= (discount / 100.0) * amount;
    System.out.println(discount + "% discount applied. Updated amount: " +
amount);
}
```

```
// Method to display order details
public void displayOrderInfo() {
    System.out.println("Order ID: " + orderId);
    System.out.println("Order Amount: " + amount);
}

public static void main(String[] args) {
    // Creating Order instances using different constructors
    Order order1 = new Order(); // Using default constructor
    Order order2 = new Order(1001); // Using constructor with orderId
    Order order3 = new Order(1002, 2000.0); // Using constructor with
    orderId and amount

    // Display order info before applying discount
    System.out.println("Order 1 (Default Constructor):");
    order1.displayOrderInfo();

    System.out.println("\nOrder 2 (Constructor with OrderId):");
    order2.displayOrderInfo();

    System.out.println("\nOrder 3 (Constructor with OrderId and Amount):");
    order3.displayOrderInfo();

    // Testing the applyDiscount method (overloading)
    System.out.println("\nApplying discounts to orders:");

    // Apply default 5% discount to order1
```

```
order1.applyDiscount();

// Apply custom 10% discount to order2
order2.applyDiscount(10);

// Apply 15% discount and extra 5% for first order to order3
order3.applyDiscount(15, true);

// Display updated order info after applying discounts
System.out.println("\nUpdated Order 1 Info:");
order1.displayOrderInfo();

System.out.println("Updated Order 2 Info:");
order2.displayOrderInfo();

System.out.println("Updated Order 3 Info:");
order3.displayOrderInfo();
}
}
```

**Output:**

Order 1 (Default Constructor):

Order ID: 0

Order Amount: 0.0

Order 2 (Constructor with OrderId):

Order ID: 1001

Order Amount: 1000.0

Order 3 (Constructor with OrderId and Amount):

Order ID: 1002

Order Amount: 2000.0

Applying discounts to orders:

5% discount applied. Updated amount: 0.0

10% discount applied. Updated amount: 900.0

First order! Adding 5% extra discount.

20% discount applied. Updated amount: 1600.0

Updated Order 1 Info:

Order ID: 0

Order Amount: 0.0

Updated Order 2 Info:

Order ID: 1001

Order Amount: 900.0

Updated Order 3 Info:

Order ID: 1002

Order Amount: 1600.0

### **Q33. Create a Class SmartDevice with Overloaded Methods:**

```
public class SmartDevice {  
    // Overloaded method 1: Control device based on device name  
    public void controlDevice(String deviceName) {  
        System.out.println("Controlling the device: " + deviceName);  
    }  
  
    // Overloaded method 2: Turn the device on or off based on boolean value  
    public void controlDevice(String deviceName, boolean turnOn) {  
        if (turnOn) {  
            System.out.println(deviceName + " is turned ON.");  
        } else {  
            System.out.println(deviceName + " is turned OFF.");  
        }  
    }  
  
    // Overloaded method 3: Adjust the device level (e.g., volume, brightness)  
    public void controlDevice(String deviceName, int level) {  
        System.out.println(deviceName + " level set to: " + level);  
    }  
  
    // Overloaded method 4: Set the mode for the device (e.g., AC modes, light  
    modes)  
    public void controlDevice(String deviceName, String mode) {  
        System.out.println(deviceName + " mode set to: " + mode);  
    }  
}
```

```
public static void main(String[] args) {  
    SmartDevice device = new SmartDevice();  
    device.controlDevice("Smart Light");  
    device.controlDevice("Smart AC", true);  
    device.controlDevice("Smart Speaker", 50);  
    device.controlDevice("Smart AC", "Cooling");  
}  
}
```

**Output:**

Controlling the device: Smart Light

Smart AC is turned ON.

Smart Speaker level set to: 50

Smart AC mode set to: Cooling

**Q34. Create a Base Class SmartDevice and Subclasses:**

// Base class: SmartDevice

```
class SmartDevice {
```

```
    // Method in base class to display generic device info
```

```
    public void deviceInfo() {
```

```
        System.out.println("Generic Smart Device");
```

```
    }
```

```
}
```

// Subclass 1: SmartLight

```
class SmartLight extends SmartDevice {  
    @Override  
    public void deviceInfo() {  
        System.out.println("Smart Light with adjustable brightness and modes.");  
    }  
}
```

// Subclass 2: SmartSpeaker

```
class SmartSpeaker extends SmartDevice {  
    @Override  
    public void deviceInfo() {  
        System.out.println("Smart Speaker with volume control and voice  
assistant.");  
    }  
}
```

// Subclass 3: SmartAC

```
class SmartAC extends SmartDevice {  
    @Override  
    public void deviceInfo() {  
        System.out.println("Smart AC with temperature control and mode  
settings.");  
    }  
}
```

// Main class to test the devices

```
public class Main {
```

```
public static void main(String[] args) {  
    // Creating objects for each subclass  
    SmartDevice device1 = new SmartDevice();  
    SmartDevice device2 = new SmartLight();  
    SmartDevice device3 = new SmartSpeaker();  
    SmartDevice device4 = new SmartAC();  
  
    // Displaying device info for each device  
    device1.deviceInfo();  
    device2.deviceInfo();  
    device3.deviceInfo();  
    device4.deviceInfo();  
}  
}
```

**Output:**

Generic Smart Device

Smart Light with adjustable brightness and modes.

Smart Speaker with volume control and voice assistant.

Smart AC with temperature control and mode settings.



### **Q35) Animal Abstract Class with Dog and Bird Subclasses:**

```
abstract class Animal {  
    String name;  
    int age;  
    String sound;  
    Animal(String name, int age, String sound) {  
        this.name = name;  
        this.age = age;  
        this.sound = sound; }  
    abstract void makeSound();  
    abstract void move();  
    void displayDetails() {  
        System.out.println("Name: " + name);  
        System.out.println("Age: " + age);  
        System.out.println("Sound: " + sound);  
    }  
}  
  
class Dog extends Animal {  
    Dog(String name, int age, String sound) {  
        super(name, age, sound);  
    } @Override  
    void makeSound() {  
        System.out.println("Dog's sound:Bark");  
    } @Override  
    void move() {  
        System.out.println("Dog is running");  
    }  
}
```

```

    }
}
class Bird extends Animal {
    Bird(String name, int age, String sound) {
        super(name, age, sound);
    } @Override
    void makeSound() {
        System.out.println("birds sound:Chirp");
    } @Override
    void move() {
        System.out.println("Birds are flying");
    }
}
public class TestAnimals {
    public static void main(String[] args) {
        Animal d = new Dog("Buddy", 3, "Bark");
        d.displayDetails();
        d.makeSound();
        d.move();
        Animal b = new Bird("Tweety", 1, "Chirp");
        b.displayDetails();
        b.makeSound();
        b.move();
    }
}

```

**Output:**

Name: Buddy

Age: 3

Sound: Bark

Dog's sound:Bark

Dog is running

Name: Tweety

Age: 1

Sound: Chirp

birds sound:Chirp

Birds are flying

**Q36) Shape Abstract Class with Circle and Rectangle Subclasses:**

```
abstract class Shape {  
    String name;  
    Shape(String name) {  
        this.name = name;  
    }  
    abstract double area();  
    abstract double perimeter();  
}  
  
class Circle extends Shape {  
    double radius;  
    Circle(double radius) {  
        super("Circle");  
        this.radius = radius;  
    }  
}
```

```
}  
  
@Override  
double area() {  
    return Math.PI * radius * radius;  
}  
  
@Override  
double perimeter() {  
    return 2 * Math.PI * radius;  
}  
}  
  
class Rectangle extends Shape {  
    double length, breadth;  
    Rectangle(double length, double breadth) {  
        super("Rectangle");  
        this.length = length;  
        this.breadth = breadth;  
    }  
    @Override  
    double area() {  
        return length * breadth;  
    }  
    @Override  
    double perimeter() {  
        return 2 * (length + breadth);  
    }  
}
```

```
public class TestShapes {  
    public static void main(String[] args) {  
        Shape circle = new Circle(5);  
        System.out.println("Circle Area: " + circle.area());  
        System.out.println("Circle Perimeter: " + circle.perimeter());  
        Shape rectangle = new Rectangle(4, 6);  
        System.out.println("Rectangle Area: " + rectangle.area());  
        System.out.println("Rectangle Perimeter: " + rectangle.perimeter());  
    }  
}
```

**Output:**

Circle Area: 78.53981633974483

Circle Perimeter: 31.41592653589793

Rectangle Area: 24.0

Rectangle Perimeter: 20.0

**Q37) Bank Account Interface with SavingAccount and Current Account Classes:**

```
interface BankAccount {  
    void deposit(double amount);  
    void withdraw(double amount);  
    double getBalance();  
}  
  
class SavingAccount implements BankAccount {  
    private double balance;
```

```
SavingAccount(double initialBalance) {
    this.balance = initialBalance;
}

@Override
public void deposit(double amount) {
    balance += amount;
    System.out.println("Deposited " + amount + " to Saving Account");
}

@Override
public void withdraw(double amount) {
    if (balance >= amount) {
        balance -= amount;
        System.out.println("Withdrew " + amount + " from Saving Account");
    } else {
        System.out.println("Insufficient funds in Saving Account");
    }
}

@Override
public double getBalance() {
    return balance;
}
}

class CurrentAccount implements BankAccount {
    private double balance;

    CurrentAccount(double initialBalance) {
        this.balance = initialBalance;
    }
}
```

```

@Override
public void deposit(double amount) {
    balance += amount;
    System.out.println("Deposited " + amount + " to Current Account");
} @Override
public void withdraw(double amount) {
    balance -= amount;
    System.out.println("Withdrew " + amount + " from Current Account");
}

@Override
public double getBalance() {
    return balance;
}
}

public class TestBankAccount {
    public static void main(String[] args) {
        BankAccount saving = new SavingAccount(1000);
        saving.deposit(500);
        saving.withdraw(200);
        System.out.println("Saving Account Balance: " + saving.getBalance());
        BankAccount current = new CurrentAccount(2000);
        current.deposit(1000);
        current.withdraw(2500);
        System.out.println("Current Account Balance: " + current.getBalance());
    }
}

```

**Output:**

Deposited 500.0 to Saving Account

Withdrew 200.0 from Saving Account

Saving Account Balance: 1300.0

Deposited 1000.0 to Current Account

Withdrew 2500.0 from Current Account

Current Account Balance: 500.0

**Q38.Vehicle interface with bike and car classes:**

```
interface Vehicle {  
    void start();  
    void stop();  
    void speed_up(int inc);  
    void break_down(int dec);  
}  
  
public class Bike implements Vehicle {  
    int speed;  
    String fuel_type;  
    String gear_sys;  
    Bike(int s,String f,String g) {  
        speed=s;  
        fuel_type=f;  
        gear_sys=g;  
    }  
    @Override  
    public void start() {
```



```
        System.out.println("Bike is running");
    } @Override
    public void stop() {
        System.out.println("Bike is stopped");
    } @Override
    public void speed_up(int inc) {
        speed=speed+inc;
    }@Override
    public void break_down(int dec) {
        speed=speed-dec;
    }
    void show_speed() {
        System.out.println("Bike speed is" + speed);
    }
    public static void main(String[] args) {
        Bike b=new Bike(40,"Petrol","Manual");
        b.start();
        b.speed_up(60);
        b.show_speed();
        b.break_down(20);
        b.show_speed();
        b.stop();
        Car c=new Car(60,"Disel","Automatic");
        c.start();
        c.speed_up(10);
        c.show_speed();
    }
}
```

```
        c.break_down(40);
        c.show_speed();
        c.stop();
    }
}

class Car implements Vehicle
{
    int speed;
    String fuel_type;
    String gear_sys;
    Car(int s,String f,String g)
    {
        speed=s;
        fuel_type=f;
        gear_sys=g;
    }
    @Override
    public void start()
    {
        System.out.println("Car is running");
    }
    @Override
    public void stop()
    {
        System.out.println("Car is stopped");
    }
}
```

```
@Override
public void speed_up(int inc)
{
    speed=speed+inc;
}

@Override
public void break_down(int dec) {
    speed=speed-dec;
}

void show_speed() {
System.out.println("Car speed is "+speed);
}
}
```

**Output:**

```
Bike is running
Bike speed is 100
Bike speed is 80
Bike is stopped
Car is running
Car speed is 70
Car speed is 30
Car is stopped
```

### **Q39. Reverse a String:**

```
public class ReverseString {  
    public static void main(String[] args) {  
        String s = "hello";  
        String reversed = "";  
        for (int i = s.length() - 1; i >= 0; i--) {  
            reversed += s.charAt(i);  
        }  
        System.out.println("Reversed String: " + reversed);  
    }  
}
```

#### **Output:**

Reversed String: olleh

### **Q40. Check if a String is a Palindrome:**

```
public class PalindromeCheck {  
    public static void main(String[] args) {  
        String s = "madam";  
        boolean isPalindrome = true;  
        for (int i = 0; i < s.length() / 2; i++) {  
            if (s.charAt(i) != s.charAt(s.length() - 1 - i)) {  
                isPalindrome = false;  
                break;  
            }  
        }  
        if (isPalindrome)
```

```

        System.out.println(s + " is a palindrome.");
    else
        System.out.println(s + " is not a palindrome.");
    }
}

```

### **Output:**

madam is a palindrome.

### **Q41. Count Vowels and Consonants:**

```

public class VowelConsonantCount {
    public static void main(String[] args) {
        String s = "hello world";
        int vowels = 0, consonants = 0;
        for (int i = 0; i < s.length(); i++) {
            char ch = Character.toLowerCase(s.charAt(i));
            if (Character.isLetter(ch)) {
                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                    vowels++;
                } else {
                    consonants++;
                }
            }
        }
        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
    }
}

```

```
}  
  
}
```

**Output:**

Vowels: 3

Consonants: 7

**Q42. Remove Duplicate Characters from a String**

```
public class RemoveDuplicates {  
    public static void main(String[] args) {  
        String s = "programming";  
        String result = "";  
        for (int i = 0; i < s.length(); i++) {  
            char ch = s.charAt(i);  
            if (result.indexOf(ch) == -1) {  
                result += ch;  
            }  
        }  
        System.out.println("String after removing duplicates: " + result);  
    }  
}
```

**Output:**

String after removing duplicates: progamin

#### **Q43. Check if Two Strings are Anagrams:**

```
public class AnagramCheck {  
    public static void main(String[] args) {  
        String s1 = "listen";  
        String s2 = "silent";  
        if (s1.length() != s2.length()) {  
            System.out.println("Not anagrams");  
            return;  
        }  
        int[] freq = new int[26];  
        for (int i = 0; i < s1.length(); i++) {  
            freq[s1.charAt(i) - 'a']++;  
            freq[s2.charAt(i) - 'a']--;  
        }  
        for (int i = 0; i < 26; i++) {  
            if (freq[i] != 0) {  
                System.out.println("Not anagrams");  
                return;  
            }  
        }  
        System.out.println("Strings are anagrams");  
    }  
}
```

#### **Output:**

Strings are anagrams

#### **Q44.Program to perform multiple risky operations like**

##### **1) Division by zero**

##### **2) Accessing a character from a null string**

##### **3) Parsing an invalid integer string**

##### **handling each exception using multiple catch blocks:**

```
import java.util.Scanner;

public class RiskyOperations {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // 1. Division operation
        try {
            System.out.print("Enter numerator: ");
            int numerator = scanner.nextInt();
            System.out.print("Enter denominator: ");
            int denominator = scanner.nextInt();
            int result = numerator / denominator;
            System.out.println("Result of division: " + result);
        } catch (ArithmeticException ae) {
            System.out.println("ArithmeticException caught: Division by zero is not allowed.");
        }

        // 2. Accessing a char from a null string
```



```

try {
    String str = null;
    char ch = str.charAt(0); // This will throw NullPointerException
    System.out.println("First character: " + ch);
}
catch (NullPointerException npe)
{
    System.out.println("NullPointerException caught: Attempted to access a
character from a null string.");
}

```

// 3. Parsing an invalid integer

```

try {
    scanner.nextLine(); // Consume leftover newline
    System.out.print("Enter a number string to parse: ");
    String input = scanner.nextLine(); // e.g., "abc"
    int parsed = Integer.parseInt(input);
    System.out.println("Parsed integer: " + parsed);
}
catch (NumberFormatException nfe){
    System.out.println("NumberFormatException caught: Invalid string
input for integer parsing.");
}

scanner.close();
System.out.println("Program has ended.");
}
}

```

### User Input:

Enter numerator: 10

Enter denominator: 0

Enter a number string to parse: abc

### Console Output:

Enter numerator: 10

Enter denominator: 0

ArithmeticException caught: Division by zero is not allowed.

NullPointerException caught: Attempted to access a character from a null string.

Enter a number string to parse: abc

NumberFormatException caught: Invalid string input for integer parsing.

Program has ended.

### **Q45.Program to Validate Voter Eligibility Using Exception Handling:**

```
class Voter {  
    void validate(int age) {  
        if (age < 18) {  
            throw new ArithmeticException("Not allowed to vote.");  
        } else {  
            System.out.println("You can vote");  
        }  
    }  
}  
  
public static void main(String[] args) {  
    Voter v = new Voter();  
    try {  
        v.validate(13);  
    }  
}
```

```
    } catch (ArithmeticException e) {  
        System.out.println(e);  
    }  
}  
}
```

Input:

v.validate(23);

output:

You can vote

java.lang.ArithmeticException: Not allowed to vote.

#### **Q46.Matrix Multiplication Program:**

```
import java.util.Scanner;  
  
public class MatrixMultiplication {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        // Input dimensions of first matrix  
        System.out.print("Enter rows of first matrix: ");  
        int r1 = sc.nextInt();  
  
        System.out.print("Enter columns of first matrix: ");  
        int c1 = sc.nextInt();  
  
        // Input dimensions of second matrix
```

```

System.out.print("Enter rows of second matrix: ");
int r2 = sc.nextInt();
System.out.print("Enter columns of second matrix: ");
int c2 = sc.nextInt();

// Check if multiplication is possible
if (c1 != r2) {
    System.out.println("Matrix multiplication not possible. Columns of first
must equal rows of second.");
    return;
}

int[][] a = new int[r1][c1];
int[][] b = new int[r2][c2];
int[][] product = new int[r1][c2];

// Input first matrix
System.out.println("Enter elements of first matrix:");
for (int i = 0; i < r1; i++)
    for (int j = 0; j < c1; j++)
        a[i][j] = sc.nextInt();

// Input second matrix
System.out.println("Enter elements of second matrix:");
for (int i = 0; i < r2; i++)
    for (int j = 0; j < c2; j++)
        b[i][j] = sc.nextInt();

```

```
// Multiply matrices
for (int i = 0; i < r1; i++) {
    for (int j = 0; j < c2; j++) {
        product[i][j] = 0;
        for (int k = 0; k < c1; k++) {
            product[i][j] += a[i][k] * b[k][j];
        }
    }
}

// Display result
System.out.println("Product of the matrices:");
for (int i = 0; i < r1; i++) {
    for (int j = 0; j < c2; j++)
        System.out.print(product[i][j] + " ");
    System.out.println();
}
}
```

Enter rows of first matrix: 2

Enter columns of first matrix: 3

Enter rows of second matrix: 4

Enter columns of second matrix: 5

Matrix multiplication not possible. Columns of first must equal rows of second.

Enter elements of first matrix:

1      2      3      4

Enter elements of second matrix:

5      6      0      7

Product of the matrices:

5      20

15    46

#### **Q47.Matrix Addition Program:**

```
import java.util.Scanner;
```

```
public class MatrixAddition {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        // Read matrix dimensions
```

```
        System.out.print("Enter number of rows: ");
```

```
        int rows = scanner.nextInt();
```

```
        System.out.print("Enter number of columns: ");
```

```
int cols = scanner.nextInt();
```

```
int[][] matrix1 = new int[rows][cols];
```

```
int[][] matrix2 = new int[rows][cols];
```

```
int[][] sum = new int[rows][cols];
```

```
// Input first matrix
```

```
System.out.println("Enter elements of first matrix:");
```

```
for (int i = 0; i < rows; i++) {
```

```
    for (int j = 0; j < cols; j++) {
```

```
        matrix1[i][j] = scanner.nextInt();
```

```
    }
```

```
}
```

```
// Input second matrix
```

```
System.out.println("Enter elements of second matrix:");
```

```
for (int i = 0; i < rows; i++) {
```

```
    for (int j = 0; j < cols; j++) {
```

```
        matrix2[i][j] = scanner.nextInt();
```

```
    }
```

```
}
```

```
// Add matrices
```

```
for (int i = 0; i < rows; i++) {
```

```
    for (int j = 0; j < cols; j++) {
```

```
        sum[i][j] = matrix1[i][j] + matrix2[i][j];
```

```
    }
```

```
}  
  
// Print result  
System.out.println("Sum of the two matrices:");  
for (int i = 0; i < rows; i++) {  
    for (int j = 0; j < cols; j++) {  
        System.out.print(sum[i][j] + " ");  
    }  
    System.out.println();  
}  
  
scanner.close();  
}  
}
```

### **Output:**

Enter number of rows: 2

Enter number of columns: 2

Enter elements of first matrix:

8      5      2      3

Enter elements of second matrix:

9      5      1      2

Sum of the two matrices:

17    10

3      5



**Q48. Write a program to display the names of all files and directories present in c:\\java:**

```
import java.io.File;
import java.io.IOException;

class FileDemo {
    public static void main(String[] args) throws IOException {
        int count = 0;
        File f = new File("C:\\java");
        String[] s = f.list();

        if (s != null) {
            for (String s1 : s) {
                count++;
                System.out.println(s1);
            }
            System.out.println("Total number: " + count);
        } else {
            System.out.println("Directory not found or is empty.");
        }
    }
}
```

### **Output:**

arcade.txt

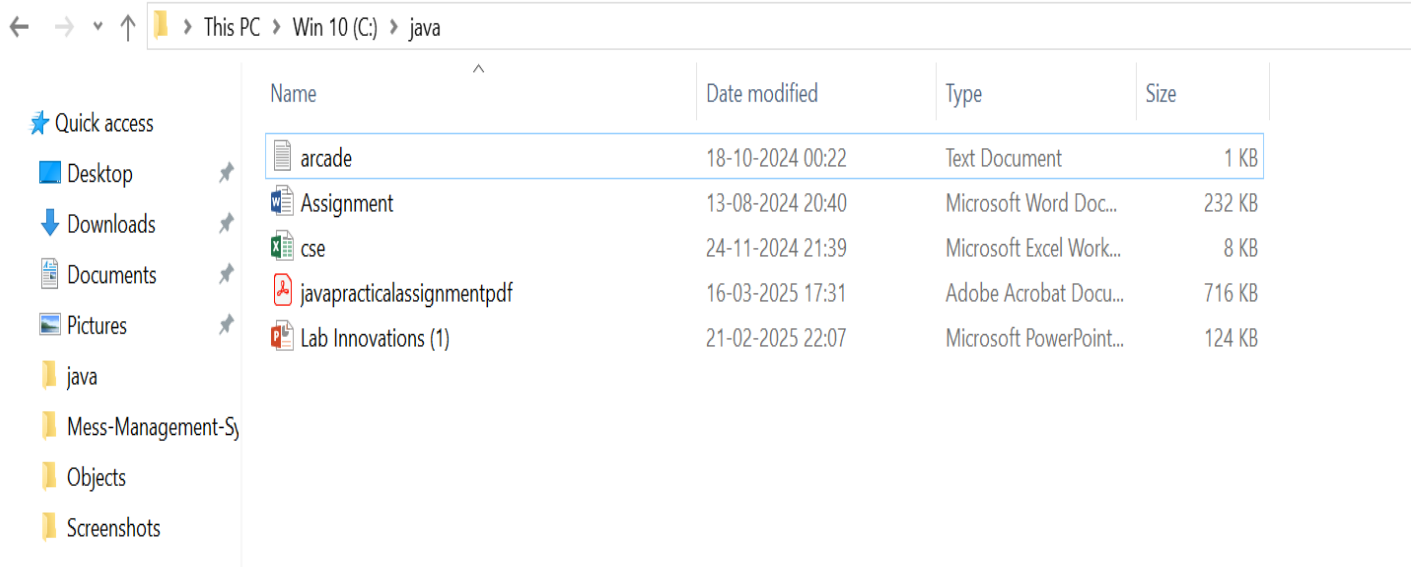
Assignment.docx

cse.xlsx

javapracticalassignmentpdf.pdf

Lab Innovations (1).pptx

Total number: 5



The screenshot shows a Windows File Explorer window with the address bar set to 'This PC > Win 10 (C:) > java'. The left sidebar shows 'Quick access' with links to Desktop, Downloads, Documents, Pictures, java, Mess-Management-Sy, Objects, and Screenshots. The main pane displays a table of files in the 'java' folder.

Name	Date modified	Type	Size
arcade	18-10-2024 00:22	Text Document	1 KB
Assignment	13-08-2024 20:40	Microsoft Word Doc...	232 KB
cse	24-11-2024 21:39	Microsoft Excel Work...	8 KB
javapracticalassignmentpdf	16-03-2025 17:31	Adobe Acrobat Docu...	716 KB
Lab Innovations (1)	21-02-2025 22:07	Microsoft PowerPoint...	124 KB

### **Q49. Write a program to display only file names :**

```
import java.io.*;

class FileDemo {

    public static void main(String[] args) throws IOException {

        int count = 0;

        File f = new File("C:\\TC");

        String[] s = f.list();
```

```

if (s != null) {
    for (String s1 : s) {
        File f1 = new File(f, s1);
        if (f1.isFile()) {
            count++;
            System.out.println(s1);
        }
    }
    System.out.println("Total number of files: " + count);
} else {
    System.out.println("Directory not found or is empty.");
}
}
}

```

### **Output:**

```

FILELIST.DOC
Like on Facebook.url
README
README.COM
turboc7.blogspot.com.url
unins000.dat
unins000.exe
Total number of files: 7

```

### **Q50.write text in file using filewriter:**

```

import java.io.*;

```

```

class FileWriterDemo {
    public static void main(String[] args) throws IOException {
        FileWriter fw = new FileWriter("C:\\java\\cricket.txt", true);
        fw.write(99); // Writes 'c'
        fw.write("Virat\\Royal Challengers Bengaluru");
        fw.write("\\n");
        String[] players = {"Rajat Patidar", "Tim David", "Krunal Pandya"};
        for (String player : players) {
            fw.write(player + " ");
        }
        fw.write("\\n");
        fw.flush();
        fw.close();
    }
}

```

← → ▾ ▴ This PC > Win 10 (C:) > TC >

	Name	Date modified	Type	Size
Objects	BGI	09-11-2024 07:55	File folder	
Screenshots	BIN	15-11-2024 09:46	File folder	
This PC	CLASSLIB	09-11-2024 07:55	File folder	
3D Objects	DOC	09-11-2024 07:55	File folder	
Desktop	EXAMPLES	09-11-2024 07:55	File folder	
Documents	INCLUDE	09-11-2024 07:56	File folder	
Downloads	LIB	09-11-2024 07:56	File folder	
Music	FILELIST	18-02-1992 03:00	Microsoft Word 97 - ...	18 KB
Pictures	Like on Facebook	01-11-2011 23:51	Internet Shortcut	1 KB
Videos	README	18-02-1992 03:00	File	16 KB
Win 10 (C:)	README	18-02-1992 03:00	MS-DOS Application	5 KB
inetpub	turboc7.blogspot.com	02-11-2011 00:13	Internet Shortcut	1 KB
Intel	unins000.dat	09-11-2024 07:56	DAT File	17 KB
java	unins000	09-11-2024 07:55	Application	768 KB

## **Output: file before execution**

cricket - Notepad

File Edit Format View Help

Cricket is a popular bat-and-ball sport played between two teams,  
typically consisting of eleven players each.  
It is known for its strategic depth, with formats ranging from  
fast-paced T20s to traditional five-day Test matches.  
cVirat\Royal Challengers Bengaluru  
Rajat Patidar Tim David Krunal Pandya

## **Output: file after execution**

### **Q51. Reading data from file:**

```
import java.io.FileReader;
```

```
import java.io.IOException;
```

```
class FileReaderDemo {
```

```
    public static void main(String[] args) throws IOException {
```

```
        FileReader fr = new FileReader("C:\\java\\cricket.txt");
```

```
        int i = fr.read();
```

```
        while (i != -1) {
```

```
            System.out.print((char) i);
```

```
            i = fr.read();
```

cricket - Notepad

File Edit Format View Help

Cricket is a popular bat-and-ball sport played between two teams,  
typically consisting of eleven players each.  
It is known for its strategic depth, with formats ranging from  
fast-paced T20s to traditional five-day Test matches.

```
}  
fr.close();  
}  
}
```

### **Output:**

Virat Kohli  
Rajat Patidar  
Devdutt Padikkal  
Swastik Chhikara  
Jitesh Sharma  
Krunal Pandya  
Manoj Bhandage  
Swapnil Singh  
Yash Dayal  
Bhuvneshwar Kumar  
Suyash Sharma  
Rasikh Dar  
Abhinandan Singh  
Mohit Rathee

Virat Kohli  
Rajat Patidar  
Devdutt Padikkal  
Swastik Chhikara  
Jitesh Sharma  
Krunal Pandya  
Manoj Bhandage  
Swapnil Singh  
Yash Dayal  
Bhuvneshwar Kumar  
Suyash Sharma  
Rasikh Dar  
Abhinandan Singh  
Mohit Rathee

### **Q52.Reading data from file using approach 2:**

```
import java.io.*;

class FileReaderDemo
{
    public static void main(String[] args)throws IOException
    { File f=new File("C:\\java\\cricket.txt");
      FileReader fr=new FileReader(f);
      char[] ch=new char[(int)f.length()];
      fr.read(ch);
      for(char ch1:ch)
      {
          System.out.print(ch1);
      }
    }
}
```

#### **Output:**

Liam Livingstone

Phil Salt

Tim David

Jacob Bethell

Romario Shepherd

Josh Hazlewood

Lungi Ngidi

Nuwan Thushara



Liam Livingstone  
Phil Salt  
Tim David  
Jacob Bethell  
Romario Shepherd  
Josh Hazlewood  
Lungi Ngidi  
Nuwan Thushara

**Q53.Writing data in file using buffered writer:**

```
import java.io.*;
```

```
class BufferedWriterDemo {
```

```
    public static void main(String[] args) throws IOException {
```

```
        FileWriter fw = new FileWriter("C:\\java\\cricket.txt ");
```

```
        BufferedWriter bw = new BufferedWriter(fw);
```

```
        bw.write(100); // writes character with ASCII value 100 ('d')
```

```
        bw.newLine();
```

```
        String[] s = {"virat", "krunal", "rajat", "Swastik"};
```

```
        for (String name : s) {
```

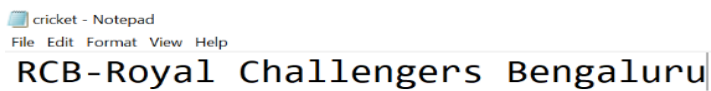
```
            bw.write(name);
```

```
            bw.newLine();
```

```
        }
```

```
bw.write("Tim David");  
  
bw.newLine();  
  
bw.write("Yash Dayal");  
  
bw.flush();  
  
bw.close();  
  
}  
  
}
```

### **Output: file before execution**



cricket - Notepad  
File Edit Format View Help  
RCB-Royal Challengers Bengaluru

### **Output: file after execution**



cricket - Notepad  
File Edit Format View Help  
d  
virat  
krunal  
rajat  
Swastik  
Tim David  
Yash Dayal

**Thank you for Reading!**

**Prepared by:**

**Sushma Kharat**

**B. Tech CSE - N.B. Navale Sinhgad  
College of Engineering, Solapur**