200+ Java Programs for Beginners

- 100% Practical
- Progressive Learning
- Easy To Follow

Hernando Abella

200+ Java Programs for Beginners

Hernando Abella

Aluna Publishing House is united by our shared passion for education, languages, and technology. Our mission is to provide the ultimate learning experience when it comes to books. We believe that books are not just words on paper; they are gateways to knowledge, imagination, and enlightenment.

Through our collective expertise, we aim to bridge the gap between traditional learning and the digital age, harnessing the power of technology to make books more accessible, interactive, and enjoyable. We are dedicated to creating a platform that fosters a love for reading, language, and lifelong learning. Join us on our journey as we embark on a quest to redefine the way you experience books.

Let's unlock the limitless potential of knowledge, one page at a time.

TABLE OF CONTENTS

INTRODUCTION

T his book is an essential tool for any beginner who wants to master Java quickly and effectively. Featuring over 200 practicals, easy-to-follow programs, this book will guide you step-by-step from the basics to more complex projects.

Each program is designed to challenge you and take your skills to the next level. Through logic exercises, fun projects, and useful applications, you will learn to think like a programmer and write clean, efficient code.

1. PRINT HELLO WORLD

```
p ublic class Main {
    public static void main(String[] args) {
    System.out.println("Hello, World!");
    }
    }
    // Hello, World!
```

2. ADD TWO NUMBERS

```
i mport java.util.Scanner ;
    public class Main {
    public static void main(String[] args) {
    // Create a scanner object for user input
    Scanner scanner = new Scanner(System.in);
    // Read numbers from user
    double num1 = readNumber(scanner, "Enter the first number: ");
    double num2 = readNumber(scanner, "Enter the second number: ");
    // Perform the addition
    double sumResult = num1 + num2;
    System.out.printf("The sum of %.2f and %.2f is: %.2f%n", num1,
num2, sumResult);
    scanner.close();
    }
    // Function to read a number from the user
    private static double readNumber(Scanner scanner, String prompt) {
    while (true) {
    System.out.print(prompt);
    String input = scanner.nextLine();
    try {
    // Try to parse the input to a double
```

```
return Double.parseDouble(input);
} catch (NumberFormatException e) {
System.out.println("Please enter a valid number.");
}
}
}
}
```

3. FIND THE SQUARE ROOT

```
i mport java.util.Scanner ;
    public class Main {
    // Function to read user input and parse it to double
    public static double readNumber(String prompt) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
    System.out.print(prompt);
    String input = scanner.nextLine();
    try {
    // Attempt to parse the input to double
    double number = Double.parseDouble(input);
    if (number >= 0) {
    return number;
    } else {
    System.out.println("Please enter a non-negative number.");
    }
    } catch (NumberFormatException e) {
    System.out.println("Please enter a valid number.");
    }
    }
    }
```

```
    public static void main(String[] args) {
    // Read a number from the user
    double inputNumber = readNumber("Enter a non-negative number: ");
    // Calculate the square root
    double squareRoot = Math.sqrt(inputNumber);
    System.out.println("The square root of " + inputNumber + " is: " +
squareRoot);
    }
    }
    // Output example:
    // Enter a non-negative number: 3
    // The square root of 3.0 is: 1.7320508075688772
```

4. CALCULATE THE AREA OF A TRIANGLE

```
i mport java.util.Scanner ;
    public class Main {
    // Function to read user input and parse it to double
    public static double readPositiveNumber(String prompt) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
    System.out.print(prompt);
    String input = scanner.nextLine();
    try {
    // Attempt to parse the input to double
    double number = Double.parseDouble(input);
    if (number > 0) {
    return number;
    } else {
    System.out.println("Please enter a valid positive number.");
    }
    } catch (NumberFormatException e) {
    System.out.println("Please enter a valid number.");
    }
    }
```

```
    }
    public static void main(String[] args) {
    // Read base and height from user
    double base = readPositiveNumber("Enter the base of the triangle: ");
    double height = readPositiveNumber("Enter the height of the triangle:
");
    // Calculate the area of the triangle
    double area = 0.5 * base * height;
    System.out.println("The area of the triangle with base " + base + " and
height " + height + " is: " + area);
    }
    }
    // Enter the base of the triangle: 3
    // Enter the height of the triangle: 4
    // The area of the triangle with base 3.0 and height 4.0 is: 6.0
```

5. SWAP TWO VARIABLES

```
i mport java.util.Scanner ;
    public class Main {
    // Function to read user input
    public static String readInput(String prompt) {
    Scanner scanner = new Scanner(System.in);
    System.out.print(prompt);
    return scanner.nextLine();
    }
    public static void main(String[] args) {
    // Read variables from user
    String variable1 = readInput("Enter the first variable: ");
    String variable2 = readInput("Enter the second variable: ");
    System.out.println("Before swapping: Variable1 = " + variable1 + ",
Variable2 = " + variable2);
    // Swapping the variables
    String temp = variable1;
    variable1 = variable2;
    variable2 = temp;
    System.out.println("After swapping: Variable1 = " + variable1 + ",
Variable2 = " + variable2);
    }
```

```
}
// Enter the first variable: a
// Enter the second variable: b
// Before swapping: Variable1 = a, Variable2 = b
// After swapping: Variable1 = b, Variable2 = a
```

6. SOLVE QUADRATIC EQUATION

```
i mport java.util.Scanner ;
    public class Main {
    // Function to read user input and parse it to double
    public static double readNumber(String prompt) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
    System.out.print(prompt);
    String input = scanner.nextLine();
    try {
    return Double.parseDouble(input);
    } catch (NumberFormatException e) {
    System.out.println("Please enter a valid number.");
    }
    }
    }
    public static void main(String[] args) {
    double a = readNumber("Enter the coefficient a: ");
    double b = readNumber("Enter the coefficient b: ");
    double c = readNumber("Enter the coefficient c: ");
    // Calculate the discriminant
    double discriminant = Math.pow(b, 2) - 4 * a * c;
```

```
if (a == 0) {
System.out.println("Coefficient 'a' cannot be zero.");
return;
}
// Check the discriminant and compute the roots
if (discriminant > 0) {
double sqrt_d = Math.sqrt(discriminant);
double root1 = (-b + sqrt_d) / (2 * a);
double root2 = (-b - sqrt_d) / (2 * a);
System.out.println("The roots are: " + root1 + " and " + root2);
} else if (discriminant == 0) {
double root = -b / (2 * a);
System.out.println("The root is: " + root);
} else {
System.out.println("The equation has complex roots.");
}
}
}
// Enter the coefficient a: 1
// Enter the coefficient b: -3
// Enter the coefficient c: 2
// The roots are: 2.0 and 1.0
```

7. CONVERT KILOMETERS TO MILES

```
i mport java.util.Scanner ;
    public class Main {
    // Function to read user input and parse it to double
    public static double readNumber(String prompt) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
    System.out.print(prompt);
    String input = scanner.nextLine();
    try {
    return Double.parseDouble(input);
    } catch (NumberFormatException e) {
    System.out.println("Please enter a valid number.");
    }
    }
    }
    public static void main(String[] args) {
    // Read distance in kilometers from user
    double kilometers = readNumber("Enter the distance in kilometers: ");
    // Conversion factor
    double kilometersToMilesConversionFactor = 0.621371;
    // Convert kilometers to miles
```

```
    double miles = kilometers * kilometersToMilesConversionFactor;
    System.out.println(kilometers + " kilometers is approximately " + miles
+ " miles.");
    }
    }
    // Enter the distance in kilometers: 2
    // 2.0 kilometers is approximately 1.242742 miles.
```

8. CONVERT CELSIUS TO FAHRENHEIT

```
i mport java.util.Scanner ;
    public class Main {
    // Function to read user input and parse it to double
    public static double readNumber(String prompt) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
    System.out.print(prompt);
    String input = scanner.nextLine();
    try {
    return Double.parseDouble(input);
    } catch (NumberFormatException e) {
    System.out.println("Please enter a valid number.");
    }
    }
    }
    public static void main(String[] args) {
    // Read temperature in Celsius from user
    double celsius = readNumber("Enter the temperature in Celsius: ");
    // Convert Celsius to Fahrenheit
    double fahrenheit = (celsius * 9.0) / 5.0 + 32.0;
```

```
        System.out.println(celsius + " degrees Celsius is equal to " + fahrenheit
+ " degrees Fahrenheit.");
        }
        }
        // Enter the temperature in Celsius: 33
        // 33.0 degrees Celsius is equal to 91.4 degrees Fahrenheit.
```

9. GENERATE A RANDOM NUMBER

```
i mport java.util.Random ;
    import java.util.Scanner;
    public class Main {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for the range
    System.out.print("Enter the minimum value of the range: ");
    double minRange = scanner.nextDouble();
    System.out.print("Enter the maximum value of the range: ");
    double maxRange = scanner.nextDouble();
    // Check if input is valid
    if (minRange < maxRange) {
    // Generate a random number within the specified range
    Random random = new Random();
    double randomNumber = minRange + (maxRange - minRange) *
random.nextDouble();
    System.out.println("A random number between " + minRange + " and "
+ maxRange + " is: " + randomNumber);
    } else {
    System.out.println("Please enter valid numbers, ensuring that the
minimum value is less than the maximum value.");
    }
```

```
}
}
// Enter the minimum value of the range: 3
// Enter the maximum value of the range: 4
// A random number between 3.0 and 4.0 is: 3.990459557841887
```

10. CHECK IF A NUMBER IS POSITIVE, NEGATIVE, OR ZERO

```
i mport java.util.Scanner ;
    public class Main {
    public static void main(String[] args) {
    // Prompt user for a number
    System.out.print("Enter a number: ");
    Scanner scanner = new Scanner(System.in);
    // Try to read the input as a floating-point number
    if (scanner.hasNextDouble()) {
    double number = scanner.nextDouble();
    // Check if the number is positive, negative, or zero
    if (number > 0.0) {
    System.out.println(number + " is a positive number.");
    } else if (number < 0.0) {
    System.out.println(number + " is a negative number.");
    } else {
    System.out.println("The entered number is zero.");
    }
    } else {
    System.out.println("Please enter a valid number.");
    }
```

```
scanner.close();
}
}
// Enter a number: 3
// 3.0 is a positive number.
```

11. CHECK IF A NUMBER IS ODD OR EVEN

```
i mport java.util.Scanner ;
    public class Main {
    public static void main(String[] args) {
    // Prompt user for a number
    System.out.print("Enter a number: ");
    Scanner scanner = new Scanner(System.in);
    // Try to read the input as an integer
    if (scanner.hasNextInt()) {
    int number = scanner.nextInt();
    // Check if the number is odd or even
    if (number % 2 == 0) {
    System.out.println(number + " is an even number.");
    } else {
    System.out.println(number + " is an odd number.");
    }
    } else {
    System.out.println("Please enter a valid integer.");
    }
    scanner.close();
    }
```

```
}
// Enter a number: 3
// 3 is an odd number.
```

12. FIND THE LARGEST AMONG THREE NUMBERS

```
i mport java.util.Scanner ;
    public class Main {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Read three numbers from the user
    double num1 = readNumber(scanner, "Enter the first number: ");
    double num2 = readNumber(scanner, "Enter the second number: ");
    double num3 = readNumber(scanner, "Enter the third number: ");
    // Find the largest number
    double largestNumber = Math.max(num1, Math.max(num2, num3));
    System.out.printf("The largest number among %.2f, %.2f, and %.2f is:
%.2f%n", num1, num2, num3, largestNumber);
    scanner.close();
    }
    // Function to read a number from user input
    private static double readNumber(Scanner scanner, String prompt) {
    System.out.print(prompt);
    while (!scanner.hasNextDouble()) {
    System.out.println("Please enter a valid number.");
    scanner.next(); // Clear invalid input
```

```
System.out.print(prompt);
}
return scanner.nextDouble();
}
}
// Enter the first number: 3
// Enter the second number: 4
// Enter the third number: 3
// The largest number among 3.00, 4.00, and 3.00 is: 4.00
```

13. CHECK PRIME NUMBER

```
i mport java.util.Scanner ;
    public class Main {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt the user to enter a number
    System.out.print("Enter a number: ");
    if (scanner.hasNextInt()) {
    int number = scanner.nextInt();
    if (number > 1 && isPrime(number)) {
    System.out.printf("%d is a prime number.%n", number);
    } else {
    System.out.printf("%d is not a prime number.%n", number);
    }
    } else {
    System.out.println("Please enter a valid integer greater than 1.");
    }
    scanner.close();
    }
    // Function to check if a number is prime
    private static boolean isPrime(int number) {
    for (int i = 2; i <= Math.sqrt(number); i++) {
```

```
if (number % i == 0) {
return false; // Found a divisor, not prime
}
}
return true; // No divisors found, is prime
}
}
// Enter a number: 33
// 33 is not a prime number.
```

14. PRINT ALL PRIME NUMBERS IN AN INTERVAL

```
i mport java.util.Scanner ;
    public class Main {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Read starting number of the interval
    System.out.print("Enter the starting number of the interval: ");
    int start = readInput(scanner);
    // Read ending number of the interval
    System.out.print("Enter the ending number of the interval: ");
    int end = readInput(scanner);
    if (start > 1 && start < end) {
    System.out.printf("Prime numbers in the interval [%d - %d]:%n", start,
end);
    for (int n = start; n <= end; n++) {
    if (isPrime(n)) {
    System.out.println(n);
    }
    }
    } else {
    System.out.println("Please enter valid integers, ensuring that the starting
number is less than the ending number and greater than 1.");
```

```
}
scanner.close();
}
// Function to read input and parse it as an integer
private static int readInput(Scanner scanner) {
while (!scanner.hasNextInt()) {
System.out.println("Please enter a valid integer.");
scanner.next(); // Clear invalid input
}
return scanner.nextInt();
}
// Function to check if a number is prime
private static boolean isPrime(int n) {
if (n <= 1) return false;
for (int i = 2; i <= Math.sqrt(n); i++) {
if (n % i == 0) {
return false; // Found a divisor, not prime
}
}
return true; // No divisors found, is prime
}
}
// Enter the starting number of the interval: 33
```

```
// Enter the ending number of the interval: 44
// Prime numbers in the interval [33 - 44]:
// 37
// 41
// 43
```

15. FIND THE FACTORIAL OF A NUMBER

```
i mport java.util.Scanner ;
    public class FactorialCalculator {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a non-negative integer: ");
    if (scanner.hasNextLong()) {
    long number = scanner.nextLong();
    System.out.println(number < 0 ? "Please enter a non-negative integer."
    : "The factorial of " + number + " is: " + calculateFactorial(number));
    } else {
    System.out.println("Please enter a valid non-negative integer.");
    }
    scanner.close();
    }
    private static long calculateFactorial(long n) {
    long result = 1;
    for (long i = 2; i <= n; i++) result *= i;
    return result;
    }
    }
```

// Enter a non-negative integer: 11

// The factorial of 11 is: 39916800

16. DISPLAY THE MULTIPLICATION TABLE

```
i mport java.util.Scanner ;
    public class MultiplicationTable {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number for the multiplication table: ");
    // Check if the input is a valid integer
    while (!scanner.hasNextInt()) {
    System.out.println("Please enter a valid integer.");
    scanner.next(); // Clear invalid input
    System.out.print("Enter a number for the multiplication table: ");
    }
    int number = scanner.nextInt();
    int rangeVal = 10;
    System.out.printf("Multiplication table for %d (up to %d):%n", number,
rangeVal);
    // Display the multiplication table
    for (int i = 1; i <= rangeVal; i++) {
    int result = number * i;
    System.out.printf("%d x %d = %d%n", number, i, result);
    }
```

```
scanner.close();
}
}
// Enter a number for the multiplication table: 2
// Multiplication table for 2 (up to 10):
// 2 x 1 = 2
// 2 x 2 = 4
// 2 x 3 = 6
// 2 x 4 = 8
// 2 x 5 = 10
// 2 x 6 = 12
// 2 x 7 = 14
// 2 x 8 = 16
// 2 x 9 = 18
// 2 x 10 = 20
```

17. PRINT THE FIBONACCI SEQUENCE

```
i mport java.util.Scanner ;
    public class FibonacciSequence {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the number of terms in the Fibonacci sequence:
");
    // Validate input
    while (!scanner.hasNextInt()) {
    System.out.println("Please enter a valid positive integer.");
    scanner.next(); // Clear invalid input
    System.out.print("Enter the number of terms in the Fibonacci sequence:
");
    }
    int numTerms = scanner.nextInt();
    if (numTerms <= 0) {
    System.out.println("Please enter a positive integer.");
    scanner.close();
    return;
    }
    // Generate Fibonacci sequence
    long[] fib = new long[numTerms];
```

```
if (numTerms > 0) fib[0] = 0; // First term
if (numTerms > 1) fib[1] = 1; // Second term
for (int i = 2; i < numTerms; i++) {
fib[i] = fib[i - 1] + fib[i - 2]; // Fibonacci formula
}
// Print the Fibonacci sequence
System.out.print("Fibonacci Sequence: ");
for (int i = 0; i < numTerms; i++) {
System.out.print(fib[i]);
if (i < numTerms - 1) {
System.out.print(", ");
}
}
System.out.println();
scanner.close();
}
}
// Enter the number of terms in the Fibonacci sequence: 5
// Fibonacci Sequence: 0, 1, 1, 2, 3
```

18. CHECK ARMSTRONG NUMBER

```
i mport java.util.Scanner ;
    public class ArmstrongNumber {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number: ");
    // Validate input
    while (!scanner.hasNextInt()) {
    System.out.println("Please enter a valid positive integer.");
    scanner.next(); // Clear invalid input
    System.out.print("Enter a number: ");
    }
    int number = scanner.nextInt();
    if (number < 0) {
    System.out.println("Please enter a valid positive integer.");
    scanner.close();
    return;
    }
    int digits = Integer.toString(number).length(); // Number of digits
    int sum = 0;
    int temp = number;
    // Calculate the sum of the powers of its digits
```

```
    while (temp > 0) {
    int digit = temp % 10; // Get last digit
    sum += Math.pow(digit, digits); // Add digit raised to the power of
digits
    temp /= 10; // Remove last digit
    }
    // Check if the sum is equal to the original number
    if (sum == number) {
    System.out.println(number + " is an Armstrong number.");
    } else {
    System.out.println(number + " is not an Armstrong number.");
    }
    scanner.close();
    }
    }
    // Enter a number: 153
    // 153 is an Armstrong number.
```

19. FIND ARMSTRONG NUMBER IN AN INTERVAL

```
i mport java.util.Scanner ;
    public class ArmstrongInInterval {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Read interval
    System.out.print("Enter the starting number: ");
    int start = getValidInteger(scanner);
    System.out.print("Enter the ending number: ");
    int end = getValidInteger(scanner);
    if (start > 0 && start < end) {
    System.out.println("Armstrong numbers in the interval [" + start + ", " +
end + "]:");
    boolean found = false;
    for (int i = start; i <= end; i++) {
    if (isArmstrong(i)) {
    System.out.println(i);
    found = true;
    }
    }
    if (!found) {
```

```
System.out.println("No Armstrong numbers found in the interval.");
}
} else {
System.out.println("Please enter valid positive integers with the starting number less than the ending number.");
}
scanner.close();
}
// Function to validate input and return an integer
private static int getValidInteger(Scanner scanner) {
while (!scanner.hasNextInt()) {
System.out.println("Please enter a valid positive integer.");
scanner.next(); // Clear invalid input
}
return scanner.nextInt();
}
// Function to check if a number is an Armstrong number
private static boolean isArmstrong(int num) {
int originalNum = num;
int sum = 0;
int digits = Integer.toString(num).length();
// Calculate the sum of the digits raised to the power of the number of digits
```

```
while (num > 0) {
int digit = num % 10;
sum += Math.pow(digit, digits);
num /= 10;
}
return sum == originalNum;
}
}
// Enter the starting number: 10
// Enter the ending number: 100
// Armstrong numbers in the interval [10, 100]:
// 10
// 11
// 12
// ...
// No Armstrong numbers found in the interval.
```

20. MAKE A SIMPLE CALCULATOR

```
i mport java.util.Scanner ;
    public class SimpleCalculator {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    double num1 = readNumber(scanner, "Enter the first number: ");
    double num2 = readNumber(scanner, "Enter the second number: ");
    String operation = readOperation(scanner);
    double result = performCalculation(num1, num2, operation);
    if (!Double.isNaN(result)) {
    System.out.printf("Result: %.2f %s %.2f = %.2f%n", num1, operation,
num2, result);
    }
    scanner.close();
    }
    // Function to read a number from user input
    private static double readNumber(Scanner scanner, String prompt) {
    System.out.print(prompt);
    while (!scanner.hasNextDouble()) {
    System.out.println("Invalid number. Please try again.");
    scanner.next(); // Clear invalid input
    System.out.print(prompt);
```

```
    }
    return scanner.nextDouble();
    }
    // Function to read an operation from user input
    private static String readOperation(Scanner scanner) {
    System.out.print("Enter an operation (+, -, *, /): ");
    return scanner.next().trim();
    }
    // Function to perform the calculation based on the operation
    private static double performCalculation(double num1, double num2,
String operation) {
    switch (operation) {
    case "+":
    return num1 + num2;
    case "-":
    return num1 - num2;
    case "*":
    return num1 * num2;
    case "/":
    if (num2 == 0) {
    System.out.println("Cannot divide by zero.");
    return Double.NaN; // Return NaN to indicate error
    }
```

```
return num1 / num2;
default:
System.out.println("Invalid operation.");
return Double.NaN; // Return NaN to indicate error
}
}
}
// Enter the first number: 10
// Enter the second number: 5
// Enter an operation (+, -, *, /): /
// Result: 10.00 / 5.00 = 2.00
```

21. FIND THE SUM OF NATURAL NUMBERS

```
i mport java.util.Scanner ;
    public class SumOfNaturalNumbers {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a positive integer
    System.out.print("Enter a positive integer: ");
    while (!scanner.hasNextLong()) {
    System.out.println("Please enter a valid positive integer.");
    scanner.next(); // Clear invalid input
    System.out.print("Enter a positive integer: ");
    }
    long n = scanner.nextLong();
    // Check if the number is positive
    if (n <= 0) {
    System.out.println("Please enter a valid positive integer.");
    } else {
    // Calculate the sum of natural numbers
    long sumOfNaturalNumbers = (n * (n + 1)) / 2;
    System.out.printf("The sum of natural numbers from 1 to %d is:
%d%n", n, sumOfNaturalNumbers);
```

```
}
scanner.close();
}
}
// Enter a positive integer: 12
// The sum of natural numbers from 1 to 12 is: 78
```

22. CHECK IF THE NUMBERS HAVE THE SAME LAST DIGIT

```
i mport java.util.Scanner ;
    public class SameLastDigit {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    long num1 = readNumber(scanner, "Enter the first number: ");
    long num2 = readNumber(scanner, "Enter the second number: ");
    if (Math.abs(num1 % 10) == Math.abs(num2 % 10)) {
    System.out.printf("The last digit of %d is the same as the last digit of
%d.%n", num1, num2);
    } else {
    System.out.printf("The last digit of %d is different from the last digit of
%d.%n", num1, num2);
    }
    scanner.close();
    }
    private static long readNumber(Scanner scanner, String prompt) {
    System.out.print(prompt);
    while (!scanner.hasNextLong()) {
    System.out.println("Please enter a valid integer.");
    scanner.next(); // Clear invalid input
    System.out.print(prompt);
```

```
}
return scanner.nextLong();
}
}
// Enter the first number: 2
// Enter the second number: 3
// The last digit of 2 is different from the last digit of 3.
```

23. FIND HCF OR GCD

```
i mport java.util.Scanner ;
    public class HCF {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    long num1 = readPositiveNumber(scanner, "Enter the first positive
integer: ");
    long num2 = readPositiveNumber(scanner, "Enter the second positive
integer: ");
    System.out.printf("The HCF (GCD) of %d and %d is: %d%n", num1,
num2, gcd(num1, num2));
    scanner.close();
    }
    private static long readPositiveNumber(Scanner scanner, String prompt)
{
    System.out.print(prompt);
    while (true) {
    if (scanner.hasNextLong()) {
    long number = scanner.nextLong();
    if (number > 0) {
    return number;
    } else {
    System.out.println("Please enter a valid positive integer.");
```

```
}
} else {
System.out.println("Please enter a valid positive integer.");
scanner.next(); // Clear invalid input
}
System.out.print(prompt); // Prompt again
}
}
private static long gcd(long a, long b) {
while (b != 0) {
long temp = b;
b = a % b;
a = temp;
}
return a;
}
}
// Enter the first positive integer: 22
// Enter the second positive integer: 33
// The HCF (GCD) of 22 and 33 is: 11
```

24. FIND LCM

```
i mport java.util.Scanner ;
    public class LCM {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    long num1 = readPositiveNumber(scanner, "Enter the first positive
integer: ");
    long num2 = readPositiveNumber(scanner, "Enter the second positive
integer: ");
    long lcm = (num1 * num2) / gcd(num1, num2);
    System.out.printf("The LCM of %d and %d is: %d%n", num1, num2,
lcm);
    scanner.close();
    }
    private static long readPositiveNumber(Scanner scanner, String prompt)
{
    System.out.print(prompt);
    while (true) {
    if (scanner.hasNextLong()) {
    long number = scanner.nextLong();
    if (number > 0) {
    return number;
    } else {
```

```
System.out.println("Please enter a valid positive integer.");
}
} else {
System.out.println("Please enter a valid positive integer.");
scanner.next(); // Clear invalid input
}
System.out.print(prompt); // Prompt again
}
}
private static long gcd(long a, long b) {
while (b != 0) {
long temp = b;
b = a % b;
a = temp;
}
return a;
}
}
// Enter the first positive integer: 22
// Enter the second positive integer: 22
// The LCM of 22 and 22 is: 22
```

25. FIND THE FACTORS OF A NUMBER

```
i mport java.util.Scanner ;
    public class Factors {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    long number = readPositiveNumber(scanner, "Enter a positive integer:
");
    System.out.printf("Factors of %d:%n", number);
    findFactors(number);
    scanner.close();
    }
    private static long readPositiveNumber(Scanner scanner, String prompt)
{
    System.out.print(prompt);
    while (true) {
    if (scanner.hasNextLong()) {
    long number = scanner.nextLong();
    if (number > 0) {
    return number;
    } else {
    System.out.println("Please enter a valid positive integer.");
    }
```

```
} else {
System.out.println("Please enter a valid positive integer.");
scanner.next(); // Clear invalid input
}
System.out.print(prompt); // Prompt again
}
}
private static void findFactors(long number) {
for (long i = 1; i <= number; i++) {
if (number % i == 0) {
System.out.println(i);
}
}
}
}
// Enter a positive integer: 44
// Factors of 44:
// 1
// 2
// 4
// 11
// 22
// 44
```

26. FIND SUM OF NATURAL NUMBERS USING RECURSION

```
i mport java.util.Scanner ;
    public class SumOfNaturalNumbers {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    long number = readPositiveNumber(scanner, "Enter a positive integer:
");
    // Display the sum using recursion
    System.out.printf("The sum of natural numbers up to %d is: %d%n",
number, sumOfNaturalNumbers(number));
    scanner.close();
    }
    // Recursive function to calculate the sum of natural numbers
    private static long sumOfNaturalNumbers(long n) {
    if (n == 0) {
    return 0;
    } else {
    return n + sumOfNaturalNumbers(n - 1);
    }
    }
    private static long readPositiveNumber(Scanner scanner, String prompt)
{
```

```
System.out.print(prompt);
while (true) {
if (scanner.hasNextLong()) {
long number = scanner.nextLong();
if (number > 0) {
return number;
} else {
System.out.println("Please enter a valid positive integer.");
}
} else {
System.out.println("Please enter a valid positive integer.");
scanner.next(); // Clear invalid input
}
System.out.print(prompt); // Prompt again
}
}
}
// Enter a positive integer: 33
// The sum of natural numbers up to 33 is: 561
```

27. GUESS A RANDOM NUMBER

```
i mport java.util.Random ;
    import java.util.Scanner;
    public class GuessRandomNumber {
    public static void main(String[] args) {
    Random random = new Random();
    int randomNumber = random.nextInt(100) + 1; // Generate a random number between 1 and 100
    int attempts = 0;
    Scanner scanner = new Scanner(System.in);
    while (true) {
    Integer guess = getGuess(scanner);
    if (guess == null) {
    System.out.println("Please enter a valid number.");
    continue;
    }
    attempts++;
    if (guess < randomNumber) {
    System.out.println("Too low!");
    } else if (guess > randomNumber) {
    System.out.println("Too high!");
    } else {
```

```
    System.out.printf("Congratulations! You guessed %d in %d attempts.%n", randomNumber, attempts);
    break;
    }
    }
    scanner.close();
    }
    private static Integer getGuess(Scanner scanner) {
    System.out.print("Guess the random number (1-100): ");
    if (scanner.hasNextInt()) {
    return scanner.nextInt();
    } else {
    scanner.next(); // Clear invalid input
    return null;
    }
    }
    }
    // Guess the random number (1-100): 50
    // Too low!
    // Guess the random number (1-100): 75
    // Too high!
    // Guess the random number (1-100): 62
    // Congratulations! You guessed 62 in 3 attempts.
```

28. SHUFFLE DECK OF CARDS

```
i mport java.util.ArrayList ;
    import java.util.Collections;
    import java.util.List;
    public class ShuffleDeckOfCards {
    public static void main(String[] args) {
    List<String> deck = createDeck();
    System.out.println("Initial Deck:");
    printDeck(deck);
    Collections.shuffle(deck); // Shuffle the deck
    System.out.println("\nShuffled Deck:");
    printDeck(deck);
    }
    private static List<String> createDeck() {
    String[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};
    String[] ranks = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack",
"Queen", "King", "Ace"};
    List<String> deck = new ArrayList<>();
    for (String suit : suits) {
    for (String rank : ranks) {
    deck.add(rank + " of " + suit);
    }
```

```
}
return deck;
}
private static void printDeck(List<String> deck) {
for (String card : deck) {
System.out.println(card);
}
}
}
// Initial Deck:
// 2 of Hearts
// 3 of Hearts
// 4 of Hearts
// 5 of Hearts
// ...
// Shuffled Deck:
// 10 of Spades
// 4 of Hearts
// Ace of Diamonds
// ...
```

29. DISPLAY FIBONACCI SEQUENCE USING RECURSION

```
i mport java.util.Scanner ;
    public class FibonacciSequence {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the number of terms in the Fibonacci sequence:
");
    int numTerms = scanner.nextInt();
    if (numTerms < 0) {
    System.out.println("Please enter a valid non-negative integer.");
    return;
    }
    System.out.println("Fibonacci sequence of " + numTerms + " terms:");
    for (int i = 0; i < numTerms; i++) {
    System.out.println(fibonacci(i));
    }
    }
    private static int fibonacci(int n) {
    if (n <= 1) {
    return n;
    } else {
```

```
return fibonacci(n - 1) + fibonacci(n - 2);
}
}
}
// Enter the number of terms in the Fibonacci sequence: 5
// Fibonacci sequence of 5 terms:
// 0
// 1
// 1
```

30. FIND FACTORIAL OF NUMBER USING RECURSION

```
i mport java.math.BigInteger ;
    import java.util.Scanner;
    public class FactorialCalculator {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a non-negative integer: ");
    int number = scanner.nextInt();
    if (number < 0) {
    System.out.println("Please enter a valid non-negative integer.");
    return;
    }
    BigInteger result = factorial(number);
    System.out.println("The factorial of " + number + " is: " + result);
    }
    private static BigInteger factorial(int n) {
    if (n == 0 || n == 1) {
    return BigInteger.ONE;
    } else {
    return BigInteger.valueOf(n).multiply(factorial(n - 1));
    }
```

```
}
}
// Enter a non-negative integer: 22
// The factorial of 22 is: 1124000727777607680000
```

31. CONVERT DECIMAL TO BINARY

```
i mport java.util.Scanner ;
    public class DecimalToBinaryConverter {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a decimal number: ");
    int decimalNumber = scanner.nextInt();
    if (decimalNumber < 0) {
    System.out.println("Please enter a valid non-negative integer.");
    return;
    }
    String binaryEquivalent = decimalToBinary(decimalNumber);
    System.out.println("The binary equivalent of " + decimalNumber + " is:
" + binaryEquivalent);
    }
    private static String decimalToBinary(int num) {
    if (num == 0) {
    return "0";
    }
    StringBuilder binary = new StringBuilder();
    int n = num;
    while (n > 0) {
```

```
binary.insert(0, n % 2);
n /= 2;
}
return binary.toString();
}
}
// Enter a decimal number: 10
// The binary equivalent of 10 is: 1010
```

32. FIND ASCII VALUE OF CHARACTER

```
i mport java.util.Scanner ;
    public class AsciiValueFinder {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a character: ");
    String input = scanner.nextLine().trim();
    if (input.length() == 1) {
    char character = input.charAt(0);
    int asciiValue = (int) character; // Convert character to ASCII value
    System.out.println("The ASCII value of '" + character + "' is: " +
asciiValue);
    } else {
    System.out.println("Please enter a valid single character.");
    }
    scanner.close();
    }
    }
    // Enter a character: a
    // The ASCII value of 'a' is: 97
```

33. CHECK WHETHER A STRING IS PALINDROME OR NOT

```
i mport java.util.Scanner ;
    public class PalindromeChecker {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a string: ");
    String input = scanner.nextLine().trim();
    String cleanedString = cleanString(input);
    if (isPalindrome(cleanedString)) {
    System.out.println("\"" + input + "\" is a palindrome.");
    } else {
    System.out.println("\"" + input + "\" is not a palindrome.");
    }
    scanner.close();
    }
    // Function to clean the string by removing non-alphanumeric characters and converting to lowercase
    private static String cleanString(String str) {
    StringBuilder cleaned = new StringBuilder();
    for (char c : str.toCharArray()) {
    if (Character.isLetterOrDigit(c)) {
```

```
cleaned.append(Character.toLowerCase(c));
}
}
return cleaned.toString();
}
// Function to check if a string is a palindrome
private static boolean isPalindrome(String s) {
String reversed = new StringBuilder(s).reverse().toString();
return s.equals(reversed);
}
}
// Enter a string: A man, a plan, a canal, Panama!
// "A man, a plan, a canal, Panama!" is a palindrome.
```

34. SORT WORDS IN ALPHABETICAL ORDER

```
i mport java.util.Arrays ;
    import java.util.Scanner;
    public class SortWords {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a sentence or a list of words
    System.out.print("Enter a sentence or a list of words: ");
    String input = scanner.nextLine().trim();
    // Split the input into an array of words
    String[] wordsList = input.split("\\s+");
    // Sort the array of words in alphabetical order
    Arrays.sort(wordsList);
    // Display the sorted words
    System.out.println("Sorted Words:");
    System.out.println(String.join(", ", wordsList));
    scanner.close();
    }
    }
    // Enter a sentence or a list of words: ad d dwqdwq qw qwd sadasd
wqdwq
```

// Sorted Words:

// ad, d, dwqdwq, qw, qwd, sadasd, wqdwq

35. REPLACE CHARACTERS OF A STRING

```
i mport java.util.Scanner ;
    public class CharacterReplacement {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a string
    System.out.print("Enter a string: ");
    String inputString = scanner.nextLine().trim();
    if (inputString.isEmpty()) {
    System.out.println("Please enter a valid string.");
    return;
    }
    // Get target and replacement characters
    String targetChar = promptUser(scanner, "Enter the target character: ");
    String replacementChar = promptUser(scanner, "Enter the replacement
character: ");
    if (targetChar.length() == 1 && replacementChar.length() == 1) {
    // Replace occurrences of targetChar with replacementChar in
inputString
    String modifiedString = inputString.replace(targetChar,
replacementChar);
    System.out.println("Modified String: " + modifiedString);
```

```
    } else {
    System.out.println("Please enter valid single characters for target and replacement.");
    }
    scanner.close();
    }
    private static String promptUser(Scanner scanner, String message) {
    System.out.print(message);
    return scanner.nextLine().trim();
    }
    }
    /*
    Enter a string: Hello World!
    Enter the target character: o
    Enter the replacement character: a
    Modified String: Hella Warld!
    */
```

36. REVERSE A STRING

```
i mport java.util.Scanner ;
    public class ReverseString {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a string
    System.out.print("Enter a string: ");
    String inputString = scanner.nextLine().trim(); // Remove trailing
newline
    // Check if input is a valid string
    if (!inputString.isEmpty()) {
    // Reverse the string
    String reversedString = new
StringBuilder(inputString).reverse().toString();
    // Display the reversed string
    System.out.println("Reversed String: " + reversedString);
    } else {
    System.out.println("Please enter a valid string.");
    }
    scanner.close();
    }
    }
```

/*

Enter a string: Hello, World!

Reversed String: !dlroW ,olleH

*/

37. CHECK THE NUMBER OF OCCURRENCES OF A CHARACTER IN THE STRING

```
i mport java.util.Scanner ;
    public class CountCharacterOccurrences {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a string
    System.out.print("Enter a string: ");
    String inputString = scanner.nextLine().trim(); // Remove trailing
newline
    // Prompt user for a character to count
    System.out.print("Enter the character to count: ");
    String targetChar = scanner.nextLine().trim(); // Remove trailing
newline
    // Check if inputs are valid
    if (!inputString.isEmpty() && targetChar.length() == 1) {
    // Count occurrences of the target character
    long count = inputString.chars()
    .filter(c -> c == targetChar.charAt(0))
    .count();
    // Display the result
```

```
    System.out.printf("Number of occurrences of '%s' in '%s': %d%n",
targetChar, inputString, count);
    } else {
    System.out.println("Please enter a valid string and a single character.");
    }
    scanner.close();
    }
    }
    /*
    Enter a string: Hello, World!
    Enter the character to count: o
    Number of occurrences of 'o' in 'Hello, World!': 2
    */
```

38. CONVERT THE FIRST LETTER OF A STRING INTO UPPERCASE

```
i mport java.util.Scanner ;
    public class CapitalizeFirstLetter {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a string
    System.out.print("Enter a string: ");
    String inputString = scanner.nextLine().trim(); // Remove trailing
newline
    // Check if input is a valid string
    if (!inputString.isEmpty()) {
    // Capitalize the first letter
    String resultString = capitalizeFirstLetter(inputString);
    // Display the result
    System.out.println("Original String: " + inputString);
    System.out.println("String with First Letter Uppercase: " +
resultString);
    } else {
    System.out.println("Please enter a valid string.");
    }
    scanner.close();
    }
```

```
// Function to capitalize the first letter of a string
private static String capitalizeFirstLetter(String s) {
if (s.isEmpty()) {
return s; // Return the original string if it's empty
}
// Capitalize the first character and append the rest of the string
return s.substring(0, 1).toUpperCase() + s.substring(1);
}
}
/*
Enter a string: hello world
Original String: hello world
String with First Letter Uppercase: Hello world
*/
```

39. COUNT THE NUMBER OF VOWELS IN A STRING

```
i mport java.util.Scanner ;
    public class CountVowels {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a string
    System.out.print("Enter a string: ");
    String inputString = scanner.nextLine().trim(); // Remove trailing
newline
    // Check if input is a valid string
    if (!inputString.isEmpty()) {
    // Call the function and display the result
    int numberOfVowels = countVowels(inputString);
    System.out.println("Number of vowels in '" + inputString + "': " +
numberOfVowels);
    } else {
    System.out.println("Please enter a valid string.");
    }
    scanner.close();
    }
    // Function to count vowels in a string
    private static int countVowels(String s) {
```

```
String vowels = "aeiouAEIOU";
return (int) s.chars()
.filter(c -> vowels.indexOf(c) != -1) // Check if the character is a vowel
.count();
}
}
/*
Enter a string: asd
Number of vowels in 'asd': 1
*/
```

40. CHECK WHETHER A STRING STARTS AND ENDS WITH CERTAIN CHARACTERS

```
i mport java.util.Scanner ;
    public class CheckStartAndEnd {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Prompt user for a string
    System.out.print("Enter a string: ");
    String inputString = scanner.nextLine().trim(); // Remove trailing
newline
    // Check if input is a valid string
    if (inputString.isEmpty()) {
    System.out.println("Please enter a valid string.");
    return;
    }
    String startChar = promptUser("Enter the starting characters: ");
    String endChar = promptUser("Enter the ending characters: ");
    // Check if the string starts and ends with the specified characters
    if (inputString.startsWith(startChar) &&
inputString.endsWith(endChar)) {
    System.out.printf("The string '%s' starts with '%s' and ends with
'%s'.%n", inputString, startChar, endChar);
```

```
    } else {
    System.out.printf("The string '%s' does not start with '%s' or end with '%s'.%n", inputString, startChar, endChar);
    }
    scanner.close();
    }
    // Method to prompt user for input
    private static String promptUser(String message) {
    System.out.print(message);
    Scanner scanner = new Scanner(System.in);
    return scanner.nextLine().trim();
    }
    }
    /*
    Enter a string: asd
    Enter the starting characters: a
    Enter the ending characters: d
    The string 'asd' starts with 'a' and ends with 'd'.
    */
```

41. REPLACE ALL OCCURRENCES OF A STRING

```
p ublic class ReplaceOccurrences {
    public static void main(String[] args) {
    // Example string
    String originalString = "Hello world, world!";
    // String to replace
    String searchString = "world";
    // Replacement string
    String replacementString = "universe";
    // Replace all occurrences of searchString with replacementString
    String modifiedString = originalString.replace(searchString,
replacementString);
    // Display the result
    System.out.println("Original String: " + originalString);
    System.out.println("Modified String: " + modifiedString);
    }
    }
    /*
    Original String: Hello world, world!
    Modified String: Hello universe, universe!
    */
```

42. CREATE MULTILINE STRINGS

```
p ublic class MultilineString {
    public static void main(String[] args) {
    // Multiline string using triple quotes (Java 13 and later)
    String multilineString = """
    This is a multiline string.
    It spans multiple lines.
    You can include line breaks and indentation easily.
    """;
    System.out.println(multilineString);
    }
    }
    /*
    This is a multiline string.
    It spans multiple lines.
    You can include line breaks and indentation easily.
    */
```

43. FORMAT NUMBERS AS CURRENCY STRINGS

```
i mport java.text.DecimalFormat ;
    public class CurrencyFormatter {
    public static void main(String[] args) {
    // Example number
    double amount = 1234567.89;
    // Format as currency string with thousands separators
    String formattedAmount = formatCurrency(amount);
    // Display the formatted currency string
    System.out.println("Formatted Amount: " + formattedAmount);
    }
    // Function to format the number as a currency string
    public static String formatCurrency(double amount) {
    // Create a DecimalFormat instance for currency formatting
    DecimalFormat formatter = new DecimalFormat("$#,###.00");
    return formatter.format(amount);
    }
    }
    /*
    Formatted Amount: $1,234,567.89
    */
```

44. GENERATE RANDOM STRING

```
i mport java.security .SecureRandom;
    public class RandomStringGenerator {
    private static final String ALPHANUMERIC =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0
123456789";
    private static final int RANDOM_STRING_LENGTH = 8;
    private static final SecureRandom random = new SecureRandom();
    public static void main(String[] args) {
    // Generate a random string of length 8
    String randomString =
generateRandomString(RANDOM_STRING_LENGTH);
    // Display the random string
    System.out.println("Random String: " + randomString);
    }
    // Function to generate a random string of a given length
    public static String generateRandomString(int length) {
    StringBuilder sb = new StringBuilder(length);
    for (int i = 0; i < length; i++) {
    int index = random.nextInt(ALPHANUMERIC.length());
    sb.append(ALPHANUMERIC.charAt(index));
    }
    return sb.toString();
```

```
}
}
/*
Random String: jQmNYWWC
*/
```

45. CHECK IF A STRING STARTS WITH ANOTHER STRING

```
p ublic class StringStartsWith {
    public static void main(String[] args) {
    // Example strings
    String mainString = "Hello, World!";
    String searchString = "Hello";
    // Check if mainString starts with searchString
    boolean startsWith = mainString.startsWith(searchString);
    // Display the result
    System.out.println("Does the string start with '" + searchString + "'? " +
startsWith);
    }
    }
    /*
    Does the string start with 'Hello'? true
    */
```

46. TRIM A STRING

```
p ublic class TrimString {
    public static void main(String[] args) {
    // Example string with leading and trailing whitespaces
    String stringWithSpaces = "   Hello, World! ";
    // Trim the string
    String trimmedString = stringWithSpaces.trim();
    // Display the result
    System.out.println("Original String: '" + stringWithSpaces + "'");
    System.out.println("Trimmed String: '" + trimmedString + "'");
    }
    }
    /*
    Original String: '   Hello, World! '
    Trimmed String: 'Hello, World!'
    */
```

47. CHECK WHETHER A STRING CONTAINS A SUBSTRING

```
p ublic class ContainsSubstring {
    public static void main(String[] args) {
    // Example string
    String mainString = "Hello, World!";
    String substringToCheck = "World";
    // Check if mainString contains substringToCheck
    boolean containsSubstring = mainString.contains(substringToCheck);
    // Display the result
    System.out.println("Does the string contain '" + substringToCheck + "'?
" + containsSubstring);
    }
    }
    /*
    Does the string contain 'World'? true
    */
```

48. COMPARE TWO STRINGS

```
p ublic class CompareStrings {
    public static void main(String[] args) {
    // Example strings
    String string1 = "Hello";
    String string2 = "hello";
    // Case-sensitive comparison
    boolean caseSensitiveComparison = string1.equals(string2);
    // Case-insensitive comparison
    boolean caseInsensitiveComparison =
string1.equalsIgnoreCase(string2);
    // Display the results
    System.out.println("Case-sensitive comparison: " +
caseSensitiveComparison);
    System.out.println("Case-insensitive comparison: " +
caseInsensitiveComparison);
    }
    }
    /*
    Case-sensitive comparison: false
    Case-insensitive comparison: true
    */
```

49. ENCODE A STRING TO BASE64

```
i mport java.util.Base64 ;
    public class EncodeStringToBase64 {
    public static void main(String[] args) {
    // Original string
    String originalString = "Hello, 你好!";
    // Encode the string to Base64
    String base64EncodedString =
Base64.getEncoder().encodeToString(originalString.getBytes());
    // Display the result
    System.out.println("Original String: " + originalString);
    System.out.println("Base64 Encoded String: " + base64EncodedString);
    }
    }
    /*
    Original String: Hello, 你好!
    Base64 Encoded String: SGVsbG8sIOS9oOWlvSE=
    */
```

50. REPLACE ALL INSTANCES OF A CHARACTER IN A STRING

```
p ublic class ReplaceCharacterInString {
    public static void main(String[] args) {
    // Example string
    String originalString = "Hello, World!";
    // Character to replace
    char charToReplace = 'l';
    // Replacement character
    char replacementChar = 'x';
    // Replace all instances of charToReplace with replacementChar
    String modifiedString = originalString.replace(charToReplace,
replacementChar);
    // Display the result
    System.out.println("Original String: " + originalString);
    System.out.println("Modified String: " + modifiedString);
    }
    }
    /*
    Original String: Hello, World!
    Modified String: Hexxo, Worxd!
    */
```

51. REPLACE ALL LINE BREAKS WITH

```
p ublic class ReplaceLineBreaks {
    public static void main(String[] args) {
    // Example string with line breaks
    String stringWithLineBreaks = "Hello,\nWorld!\nThis is a new line.";
    // Replacement string
    String replacementString = "-";
    // Replace all line breaks with the replacement string
    String stringWithoutLineBreaks = stringWithLineBreaks.replace("\n",
replacementString);
    // Display the result
    System.out.println("Original String:");
    System.out.println(stringWithLineBreaks);
    System.out.println("\nString without Line Breaks:");
    System.out.println(stringWithoutLineBreaks);
    }
    }
    /*
    Original String:
    Hello,
    World!
```

This is a new line.

String without Line Breaks:

Hello,-World!-This is a new line.

*/

52. CHECK LEAP YEAR

```
p ublic class LeapYearChecker {
    // Method to check if a year is a leap year
    public static boolean isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }
    public static void main(String[] args) {
    // Example year to check
    int yearToCheck = 2024;
    // Check if the year is a leap year
    if (isLeapYear(yearToCheck)) {
    System.out.println(yearToCheck + " is a leap year.");
    } else {
    System.out.println(yearToCheck + " is not a leap year.");
    }
    }
    }
    /*
    2024 is a leap year.
    */
```

53. FORMAT THE DATE

```
i mport java.time.LocalDateTime ;
    import java.time.format.DateTimeFormatter;
    public class DateFormatter {
    public static void main(String[] args) {
    // Get the current date and time
    LocalDateTime currentDate = LocalDateTime.now();
    // Format the date
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("EEEE,
MMMM dd, yyyy");
    String formattedDate = currentDate.format(formatter);
    // Display the result
    System.out.println("Formatted Date: " + formattedDate);
    }
    }
    /*
    Formatted Date: Friday, September 20, 2024
    */
```

54. DISPLAY CURRENT DATE

```
i mport java.time.LocalDate ;
    import java.time.format.DateTimeFormatter;
    public class CurrentDate {
    public static void main(String[] args) {
    // Get the current date
    LocalDate currentDate = LocalDate.now();
    // Format the current date
    DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("MM/dd/yyyy");
    String formattedDate = currentDate.format(formatter);
    // Display the result
    System.out.println("Current Date: " + formattedDate);
    }
    }
    /*
    Current Date: 09/20/2024
    */
```

55. COMPARE THE VALUE OF TWO DATES

```
i mport java.time.LocalDate ;
    import java.time.ZoneOffset;
    public class CompareDates {
    public static void main(String[] args) {
    // Parse the example dates
    LocalDate date1 = LocalDate.of(2022, 1, 1);
    LocalDate date2 = LocalDate.of(2023, 1, 1);
    // Compare dates
    if (date1.isBefore(date2)) {
    System.out.println(date1 + " is earlier than " + date2);
    } else if (date1.isAfter(date2)) {
    System.out.println(date1 + " is later than " + date2);
    } else {
    System.out.println(date1 + " is equal to " + date2);
    }
    }
    }
    /*
    2022-01-01 is earlier than 2023-01-01
    */
```

56. CREATE COUNTDOWN TIMER

```
i mport java.time.LocalDateTime ;
    import java.time.Duration;
    public class CountdownTimer {
    public static void main(String[] args) throws InterruptedException {
    // Set the target date to 1 minute from now
    LocalDateTime targetDate = LocalDateTime.now().plusMinutes(1);
    while (true) {
    // Calculate the remaining time
    Duration timeDiff = Duration.between(LocalDateTime.now(),
targetDate);
    if (timeDiff.isNegative() || timeDiff.isZero()) {
    break; // Exit the loop if the countdown has expired
    }
    long days = timeDiff.toDays();
    long hours = timeDiff.toHours() % 24;
    long minutes = timeDiff.toMinutes() % 60;
    long seconds = timeDiff.getSeconds() % 60;
    // Display the countdown
    System.out.printf("Countdown: %dd %dh %dm %ds%n", days, hours,
minutes, seconds);
    Thread.sleep(1000); // Sleep for 1 second
```

```
}
System.out.println("Countdown expired!");
}
}
/*
Countdown: 0d 0h 0m 56s
Countdown expired!
*/
```

57. REMOVE SPECIFIC ITEM FROM AN ARRAY

```
i mport java.util.ArrayList ;
    import java.util.Arrays;
    public class RemoveItemFromArray {
    public static void main(String[] args) {
    // Example list
    ArrayList<Integer> originalList = new ArrayList<>(Arrays.asList(1, 2,
3, 4, 5));
    int itemToRemove = 3;
    // Find the position of the item to remove
    if (originalList.remove(Integer.valueOf(itemToRemove))) {
    System.out.println("Original List: " + originalList);
    } else {
    System.out.println("Item not found in the list.");
    }
    }
    }
    /*
    Original List: [1, 2, 4, 5]
    */
```

58. CHECK IF AN ARRAY CONTAINS A SPECIFIED VALUE

```
i mport java.util.Arrays ;
    public class CheckArrayContainsValue {
    public static void main(String[] args) {
    // Example array
    int[] myList = {1, 2, 3, 4, 5};
    int valueToCheck = 3;
    // Check if the array contains the specified value
    boolean containsValue = contains(myList, valueToCheck);
    // Display the result
    System.out.println("Does the list include " + valueToCheck + "? " +
containsValue);
    }
    // Method to check if the array contains the specified value
    public static boolean contains(int[] array, int value) {
    for (int num : array) {
    if (num == value) {
    return true; // Value found
    }
    }
    return false; // Value not found
```

```
}
}
/*
Does the list include 3? true
*/
```

59. INSERT ITEM IN AN ARRAY

```
i mport java.util.ArrayList ;
    public class InsertItemInArray {
    public static void main(String[] args) {
    // Example ArrayList
    ArrayList<Integer> myList = new ArrayList<>();
    myList.add(1);
    myList.add(2);
    myList.add(3);
    myList.add(4);
    myList.add(5);
    int itemToInsert = 6;
    // Insert the item at the end of the ArrayList
    myList.add(itemToInsert);
    // Display the result
    System.out.println("List after inserting: " + myList);
    }
    }
    /*
    List after inserting: [1, 2, 3, 4, 5, 6]
    */
```

60. GET RANDOM ITEM FROM AN ARRAY

```
i mport java.util.ArrayList ;
    import java.util.List;
    import java.util.Random;
    public class GetRandomItemFromArray {
    public static void main(String[] args) {
    // Example ArrayList
    List<Integer> myList = new ArrayList<>();
    for (int i = 1; i <= 10; i++) {
    myList.add(i);
    }
    // Create a random number generator
    Random random = new Random();
    // Get a random item from the ArrayList
    if (!myList.isEmpty()) {
    int randomIndex = random.nextInt(myList.size());
    int randomItem = myList.get(randomIndex);
    // Display the result
    System.out.println("Random Item: " + randomItem);
    } else {
    System.out.println("The list is empty.");
```

```
}
}
}
/*
Random Item: 4
*/
```

61. PERFORM INTERSECTION BETWEEN TWO ARRAYS

```
i mport java.util.ArrayList ;
    import java.util.HashSet;
    import java.util.List;
    import java.util.Set;
    public class ArrayIntersection {
    public static void main(String[] args) {
    // Example ArrayLists
    List<Integer> list1 = new ArrayList<>();
    List<Integer> list2 = new ArrayList<>();
    // Fill the first list
    list1.add(1);
    list1.add(2);
    list1.add(3);
    list1.add(4);
    list1.add(5);
    // Fill the second list
    list2.add(3);
    list2.add(4);
    list2.add(5);
    list2.add(6);
    list2.add(7);
```

```
// Create HashSets from the lists
Set<Integer> set1 = new HashSet<>(list1);
Set<Integer> set2 = new HashSet<>(list2);
// Find the intersection of the two HashSets
set1.retainAll(set2);
// Display the result
System.out.println("Intersection: " + set1);
}
}
/*
Intersection: [3, 4, 5]
*/
```

62. SPLIT ARRAY INTO SMALLER CHUNKS

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class ChunkArray {
    public static <T> List<List<T>> chunkArray(List<T> array, int
chunkSize) {
    List<List<T>> chunks = new ArrayList<>();
    for (int i = 0; i < array.size(); i += chunkSize) {
    int end = Math.min(i + chunkSize, array.size());
    chunks.add(new ArrayList<>(array.subList(i, end)));
    }
    return chunks;
    }
    public static void main(String[] args) {
    // Example array
    List<Integer> myArray = List.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    // Split the array into chunks of size 3
    List<List<Integer>> chunks = chunkArray(myArray, 3);
    // Display the result
    System.out.println("Original Array: " + myArray);
    System.out.println("Chunks: " + chunks);
```

```
}
}
/*
Original Array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Chunks: [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10]]
*/
```

63. GET FILE EXTENSION

```
i mport java.io.File ;
    public class FileExtension {
    public static String getFileExtension(String fileName) {
    // Create a File object from the file name
    File file = new File(fileName);
    // Get the file extension
    String extension = "";
    // Check if the file has a name
    if (file.getName().contains(".")) {
    // Extract the extension from the file name
    extension = file.getName().substring(file.getName().lastIndexOf('.') +
1);
    }
    return extension.isEmpty() ? null : extension; // Return null if no
extension is found
    }
    public static void main(String[] args) {
    // Example file name
    String fileName = "example.txt";
    // Get the file extension
    String extension = getFileExtension(fileName);
```

```
if (extension != null) {
System.out.println("File Extension: " + extension);
} else {
System.out.println("No file extension found.");
}
}
}
/*
File Extension: txt
*/
```

64. CHECK IF A VARIABLE IS UNDEFINED OR NULL

```
p ublic class VariableCheck {
    public static void main(String[] args) {
    // Example variables
    Integer undefinedVariable = null; // Using Integer wrapper to allow null
    String nullVariable = null;
    String valueVariable = "Hello, World!";
    // Check the variables
    checkVariable(undefinedVariable); // Variable is null
    checkVariable(nullVariable);       // Variable is null
    checkVariable(valueVariable);      // Variable has a value
    }
    public static void checkVariable(Object variable) {
    // Check if the variable is null
    if (variable == null) {
    System.out.println("Variable is null");
    } else {
    System.out.println("Variable has a value: " + variable);
    }
    }
    }
```

// Variable is null

// Variable is null

// Variable has a value: Hello, World!

65. GENERATE A RANDOM NUMBER BETWEEN TWO NUMBERS

```
i mport java.util.Random ;
    public class RandomNumberGenerator {
    public static void main(String[] args) {
    int minVal = 1;
    int maxVal = 100;
    int randomNum = getRandomNumber(minVal, maxVal); // Generate a random number between 1 and 100
    System.out.println("Random Number: " + randomNum);
    }
    public static int getRandomNumber(int minVal, int maxVal) {
    Random rand = new Random(); // Create a Random object
    return rand.nextInt(maxVal - minVal + 1) + minVal; // Generate a random number in the range [minVal, maxVal]
    }
    }
    // Random Number: 48
```

66. LONGEST DAILY STREAK

```
p ublic class LongestDailyStreak {
    public static void main(String[] args) {
    // Test cases
    System.out.println(dailyStreak(new boolean[]{true, true, false, true}));
// Outputs: 2
    System.out.println(dailyStreak(new boolean[]{false, false, false}));    //
Outputs: 0
    System.out.println(dailyStreak(new boolean[]{true, true, true, false,
true, true})); // Outputs: 3
    }
    public static int dailyStreak(boolean[] logins) {
    int maxStreak = 0;
    int currentStreak = 0;
    for (boolean loggedIn : logins) {
    if (loggedIn) {
    currentStreak++; // Increase current streak if logged in
    if (currentStreak > maxStreak) {
    maxStreak = currentStreak; // Update max streak if needed
    }
    } else {
    currentStreak = 0; // Reset current streak if not logged in
    }
```

```
}
return maxStreak; // Return the longest streak found
}
}
// 2
// 0
// 3
```

67. VALIDATE AN EMAIL ADDRESS

```
i mport java.util.regex .Pattern;
    import java.util.regex.Matcher;
    public class EmailValidator {
    // Method to validate the email address
    public static boolean validateEmail(String email) {
    // Regular expression for basic email validation
    String emailRegex = "^[^\\s@]+@[^\\s@]+\\.[^\\s@]+$";
    // Compile the pattern
    Pattern pattern = Pattern.compile(emailRegex);
    // Match the email against the regular expression
    Matcher matcher = pattern.matcher(email);
    return matcher.matches();
    }
    public static void main(String[] args) {
    // Example usage
    String emailToValidate = "example@email.com";
    if (validateEmail(emailToValidate)) {
    // Output: Email is valid
    System.out.println("Email is valid");
    } else {
    // Output: Email is not valid
```

```
System.out.println("Email is not valid");
}
}
}
```

68. RECORD TEMPERATURES

```
i mport java.util.Arrays ;
    public class TemperatureRecord {
    // Method to record the temperatures
    public static int[][] recordTemps(int[][] record, int[][] current) {
    for (int i = 0; i < current.length; i++) {
    int[] dailyTemps = current[i];
    // Update record low if current daily low is lower
    if (dailyTemps[0] < record[i][0]) {
    record[i][0] = dailyTemps[0];
    }
    // Update record high if current daily high is higher
    if (dailyTemps[1] > record[i][1]) {
    record[i][1] = dailyTemps[1];
    }
    }
    return record;
    }
    public static void main(String[] args) {
    // Initial record temperatures
    int[][] records = {
    {34, 82}, {24, 82}, {20, 89}, {5, 88}, {9, 88}, {26, 89}, {27, 83}
```

```
};
// Current week's temperatures
int[][] currentWeekTemps = {
{44, 72}, {19, 70}, {40, 69}, {39, 68}, {33, 64}, {36, 70}, {38, 69}
};
// Updating the record temperatures
int[][] updatedRecords = recordTemps(records, currentWeekTemps);
// Output: [[34, 82], [19, 82], [20, 89], [5, 88], [9, 88], [26, 89], [27, 83]]
System.out.println(Arrays.deepToString(updatedRecords));
}
}
```

69. KAPREKAR NUMBERS

```
p ublic class KaprekarNumber {
    // Method to check if a number is a Kaprekar number
    public static boolean isKaprekar(long n) {
    if (n == 0 || n == 1) {
    return true; // 0 and 1 are Kaprekar numbers
    }
    // Calculate the square of n
    long square = n * n;
    String squareStr = Long.toString(square);
    int len = squareStr.length();
    // Split the squared number into left and right parts
    String leftStr, rightStr;
    if (len == 1) {
    leftStr = ""; // Only right part
    rightStr = squareStr;
    } else {
    int splitIndex = len / 2;
    leftStr = squareStr.substring(0, len - splitIndex);
    rightStr = squareStr.substring(len - splitIndex);
    }
    // Convert left and right parts to numbers
```

```
long left = leftStr.isEmpty() ? 0 : Long.parseLong(leftStr);
long right = rightStr.isEmpty() ? 0 : Long.parseLong(rightStr);
// Check if the sum of left and right equals n
return left + right == n;
}
public static void main(String[] args) {
// Test cases
// Output: false
System.out.println("isKaprekar(3) ⟶ " + isKaprekar(3));
// Output: false
System.out.println("isKaprekar(5) ⟶ " + isKaprekar(5));
// Output: true
System.out.println("isKaprekar(297) ⟶ " + isKaprekar(297));
}
}
```

70. PASS PARAMETER TO A THREADING.TIMER FUNCTION

```
i mport java.util.concurrent .Executors;
    import java.util.concurrent.ScheduledExecutorService;
    import java.util.concurrent.TimeUnit;
    public class TimerExample {
    // Function that takes a parameter and prints it
    public static void myFunction(String parameter) {
    System.out.println("Parameter received: " + parameter);
    }
    public static void main(String[] args) {
    // Define the parameter
    String myParameter = "Hello, world!";
    // Create a scheduled executor service with a single thread
    ScheduledExecutorService executor =
Executors.newSingleThreadScheduledExecutor();
    // Define a task to be executed after a delay
    Runnable delayedExecution = () -> myFunction(myParameter);
    // Schedule the task with a 1-second delay
    executor.schedule(delayedExecution, 1, TimeUnit.SECONDS);
    // Shutdown the executor after the task completes
    executor.shutdown();
```

```
}
}
```

71. GENERATE A RANGE OF NUMBERS AND CHARACTERS

```
i mport java.util.List ;
    import java.util.ArrayList;
    public class RangeGenerator {
    // Method to generate a range of numbers from start to end
    public static List<Integer> generateNumberRange(int start, int end) {
    List<Integer> numberRange = new ArrayList<>();
    for (int i = start; i <= end; i++) {
    numberRange.add(i);
    }
    return numberRange;
    }
    // Method to generate a range of characters from start to end
    public static List<Character> generateCharRange(char start, char end) {
    List<Character> charRange = new ArrayList<>();
    for (char c = start; c <= end; c++) {
    charRange.add(c);
    }
    return charRange;
    }
    public static void main(String[] args) {
```

```
// Generate numbers from 1 to 5
List<Integer> numberRange = generateNumberRange(1, 5);
System.out.println("Number Range: " + numberRange);
// Generate characters from 'a' to 'e'
List<Character> charRange = generateCharRange('a', 'e');
System.out.println("Character Range: " + charRange);
}
}
```

72. PERFORM FUNCTION OVERLOADING

```
p ublic class FunctionOverloading {
    // Method with no arguments
    public static void exampleFunction() {
    System.out.println("No arguments");
    }
    // Method with one number argument
    public static void exampleFunction(int n) {
    System.out.println("One number argument: " + n);
    }
    // Method with string and number arguments
    public static void exampleFunction(String s, int n) {
    System.out.println("String and number arguments: " + s + ", " + n);
    }
    public static void main(String[] args) {
    // Calls the method with no arguments
    exampleFunction();
    // Calls the method with one integer argument
    exampleFunction(42);
    // Calls the method with a string and an integer argument
    exampleFunction("Hello", 7);
```

```
// The following line will work correctly due to method overloading:
// exampleFunction("world", 7);
}
}
```

73. REVERSE IMAGE

```
i mport java.util.Arrays ;
    public class ImageReverser {
    // Method to reverse a binary image
    public static int[][] reverseImage(int[][] image) {
    int[][] reversedImage = new int[image.length][];
    for (int i = 0; i < image.length; i++) {
    reversedImage[i] = new int[image[i].length];
    for (int j = 0; j < image[i].length; j++) {
    // Reverse pixel (1 -> 0, 0 -> 1)
    reversedImage[i][j] = 1 - image[i][j];
    }
    }
    return reversedImage;
    }
    public static void main(String[] args) {
    // Example 1
    int[][] image1 = {
    {1, 0, 0},
    {0, 1, 0},
    {0, 0, 1}
    };
```

```
int[][] reversed1 = reverseImage(image1);
System.out.println(Arrays.deepToString(reversed1));
// ➞ [[0, 1, 1], [1, 0, 1], [1, 1, 0]]
// Example 2
int[][] image2 = {
{1, 1, 1},
{0, 0, 0}
};
int[][] reversed2 = reverseImage(image2);
System.out.println(Arrays.deepToString(reversed2));
// ➞ [[0, 0, 0], [1, 1, 1]]
// Example 3
int[][] image3 = {
{1, 0, 0},
{1, 0, 0}
};
int[][] reversed3 = reverseImage(image3);
System.out.println(Arrays.deepToString(reversed3));
// ➞ [[0, 1, 1], [0, 1, 1]]
}
}
```

74. NO YELLING

```
p ublic class NoYelling {
    // Method to remove extra exclamation or question marks from the end of a sentence
    public static String noYelling(String sentence) {
    // Remove trailing '!' or '?' from the sentence
    String trimmed = sentence.replaceAll("[!?]+$", "");
    char lastChar = sentence.charAt(sentence.length() - 1);
    // Check if the last character was '!' or '?'
    if (lastChar == '!') {
    return trimmed + "!";
    } else if (lastChar == '?') {
    return trimmed + "?";
    } else {
    return sentence;
    }
    }
    public static void main(String[] args) {
    System.out.println(noYelling("What went wrong??????????"));  // ➞ "What went wrong?"
    System.out.println(noYelling("Oh my goodness!!!"));         // ➞ "Oh my goodness!"
```

```
    System.out.println(noYelling("I just!!! can!!! not!!! believe!!! it!!!")); // → "I just!!! can!!! not!!! believe!!! it!"
    System.out.println(noYelling("Oh my goodness!"));          // → "Oh my goodness!"
    System.out.println(noYelling("I just cannot believe it.")); // → "I just cannot believe it."
  }
}
```

75. CHECK IF A NUMBER IS FLOAT OR INTEGER

```
p ublic class NumberTypeChecker {
    // Method to check if a number is an integer or a float
    public static void checkNumberType(Object number) {
    if (number instanceof Integer) {
    System.out.println(number + " is an integer.");
    } else if (number instanceof Double) {
    System.out.println(number + " is a float.");
    } else if (number instanceof String) {
    try {
    // Try parsing the string as an integer first
    Integer.parseInt((String) number);
    System.out.println(number + " is an integer.");
    } catch (NumberFormatException e1) {
    try {
    // Try parsing the string as a double
    Double.parseDouble((String) number);
    System.out.println(number + " is a float.");
    } catch (NumberFormatException e2) {
    System.out.println(number + " is not a valid number.");
    }
```

```
}
} else {
System.out.println(number + " is not a valid number.");
}
}
public static void main(String[] args) {
checkNumberType(5);         // Outputs: 5 is an integer.
checkNumberType(3.14);      // Outputs: 3.14 is a float.
checkNumberType(7.0);       // Outputs: 7.0 is a float.
checkNumberType(-2.5);      // Outputs: -2.5 is a float.
checkNumberType("abc");     // Outputs: abc is not a valid number.
}
}
```

76. PASS A FUNCTION AS PARAMETER

```
i mport java.util.function .BiFunction;
    public class FunctionParameterExample {
    // Method that takes two integers and a BiFunction as a parameter
    public static int operateOnNumbers(int a, int b, BiFunction<Integer, Integer, Integer> operation) {
    return operation.apply(a, b); // Call the passed function with the provided arguments
    }
    public static void main(String[] args) {
    // Example usage with addition
    int result1 = operateOnNumbers(3, 5, (x, y) -> x + y);
    System.out.println("Result of addition: " + result1);  // Outputs: 8
    // Example usage with multiplication
    int result2 = operateOnNumbers(3, 5, (x, y) -> x * y);
    System.out.println("Result of multiplication: " + result2);  // Outputs: 15
    }
    }
```

77. SLIDEY NUMBERS

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class SlideyNumbers {
    // Method to check if a number is slidey
    public static boolean isSlidey(int n) {
    // Convert the number to a string and extract digits
    List<Integer> digits = new ArrayList<>();
    for (char c : Integer.toString(n).toCharArray()) {
    digits.add(Character.getNumericValue(c));
    }
    // All single-digit numbers are considered slidey
    if (digits.size() <= 1) {
    return true;
    }
    // Check the absolute difference between consecutive digits
    for (int i = 0; i < digits.size() - 1; i++) {
    if (Math.abs(digits.get(i) - digits.get(i + 1)) != 1) {
    return false; // Not slidey if the difference is not 1
    }
    }
    return true; // If all checks pass, it's slidey
```

```
}
public static void main(String[] args) {
// Test cases
System.out.println(isSlidey(123454321)); // Outputs: true
System.out.println(isSlidey(54345));     // Outputs: true
System.out.println(isSlidey(987654321)); // Outputs: true
System.out.println(isSlidey(1123));      // Outputs: false
System.out.println(isSlidey(1357));      // Outputs: false
}
}
```

78. REMOVE ALL WHITESPACES FROM A TEXT

```
p ublic class RemoveWhitespaces {
    public static void main(String[] args) {
    String textWithWhitespaces = "This is a text with spaces";
    String textWithoutWhitespaces =
removeWhitespaces(textWithWhitespaces);
    System.out.println("Original Text: " + textWithWhitespaces);
    System.out.println("Text without Whitespaces: " +
textWithoutWhitespaces);
    }
    public static String removeWhitespaces(String inputText) {
    // Replace all whitespace characters with an empty string
    return inputText.replaceAll("\\s", "");
    }
    }
    // Output:
    // Original Text: This is a text with spaces
    // Text without Whitespaces: Thisisatextwithspaces
```

79. WRITE TO CONSOLE

```
p ublic class ConsoleOutput {
    public static void main(String[] args) {
    // Write a message to the console
    System.out.println("Hello, world!");
    // You can also print variables or expressions
    int number = 42;
    System.out.println("The answer is: " + number);
    // Multiple values can be printed in a single statement
    String firstName = "John";
    String lastName = "Doe";
    System.out.println("Full Name: " + firstName + " " + lastName);
    }
    }
    // Output:
    // Hello, world!
    // The answer is: 42
    // Full Name: John Doe
```

80. CONVERT DATE TO NUMBER

```
p ublic class DateToNumber {
    public static void main(String[] args) {
    // Get the current time in milliseconds since Unix Epoch
    long millisecondsSinceEpoch = System.currentTimeMillis();
    // Print the numerical representation in milliseconds
    System.out.println("Current Time in milliseconds since Unix Epoch: " +
millisecondsSinceEpoch);
    }
    }
    // Output:
    // Current Time in milliseconds since Unix Epoch: 1726202472472
```

81. FIND THE AVERAGE OF TWO NUMBERS

```
p ublic class FindAverage {
    public static double findAverage(double num1, double num2) {
    // Calculate the sum of the two numbers
    double total = num1 + num2;
    // Calculate the average by dividing the sum by 2
    return total / 2.0;
    }
    public static void main(String[] args) {
    double number1 = 10.0;
    double number2 = 20.0;
    double result = findAverage(number1, number2);
    System.out.println("The average of " + number1 + " and " + number2 +
" is: " + result);
    }
    }
    // Output:
    // The average of 10.0 and 20.0 is: 15.0
```

82. CALCULATE THE AREA OF A CIRCLE

```
p ublic class CircleArea {
    public static double calculateCircleArea(double radius) throws
IllegalArgumentException {
    // Check if the radius is a valid number
    if (radius <= 0) {
    throw new IllegalArgumentException("Invalid radius. Please provide a
positive number.");
    }
    // Calculate the area
    return Math.PI * Math.pow(radius, 2);
    }
    public static void main(String[] args) {
    double radius = 5.0;
    try {
    double area = calculateCircleArea(radius);
    System.out.println("The area of a circle with radius " + radius + " is: " +
area);
    } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
    }
    }
```

```
}
// Output:
// The area of a circle with radius 5.0 is: 78.53981633974483
```

83. NUMBERED ALPHABET

```
p ublic class NumberedAlphabet {
    public static String alphNum(String s) {
    StringBuilder result = new StringBuilder();
    for (char c : s.toUpperCase().toCharArray()) {
    // Calculate the position in the alphabet
    int index = c - 'A';
    result.append(index).append(" ");
    }
    // Remove the trailing space
    return result.toString().trim();
    }
    public static void main(String[] args) {
    String[] examples = {"XYZ", "ABCDEF", "JAVASCRIPT"};
    for (String example : examples) {
    String result = alphNum(example);
    System.out.println(example + " ➞ " + result);
    }
    }
    }
    // Output:
    // XYZ ➞ 23 24 25
```

// ABCDEF ➞ 0 1 2 3 4 5

// JAVASCRIPT ➞ 9 0 21 0 18 2 17 8 15 19

84. CHECK IF A STRING IS EMPTY

```
p ublic class CheckEmptyString {
    public static boolean isEmptyString(String string) {
    return string.isEmpty();
    }
    public static void main(String[] args) {
    String emptyString = "";
    String nonEmptyString = "Hello, world!";
    System.out.println("Is emptyString empty? " +
isEmptyString(emptyString));
    System.out.println("Is nonEmptyString empty? " +
isEmptyString(nonEmptyString));
    }
    }
    // Output:
    // Is emptyString empty? true
    // Is nonEmptyString empty? false
```

85. CAPITALIZE THE FIRST LETTER OF A STRING

```
p ublic class CapitalizeFirstLetter {
    public static String capitalizeFirstLetter(String string) {
    if (string.isEmpty()) {
    return "Empty string";
    }
    char firstChar = Character.toUpperCase(string.charAt(0));
    String rest = string.substring(1);
    return firstChar + rest;
    }
    public static void main(String[] args) {
    String originalString = "hello, world!";
    String capitalizedString = capitalizeFirstLetter(originalString);
    System.out.println("Original String: " + originalString);
    System.out.println("Capitalized String: " + capitalizedString);
    }
    }
    // Output:
    // Original String: hello, world!
    // Capitalized String: Hello, world!
```

86. FIND THE MAXIMUM ELEMENT IN AN ARRAY

```
p ublic class MaxElementFinder {
    public static Integer findMaxElement(int[] arr) {
    // Check if the array is not empty
    if (arr.length == 0) {
    return null; // Return null if the array is empty
    }
    int max = arr[0]; // Initialize max with the first element
    for (int num : arr) {
    if (num > max) {
    max = num; // Update max if a larger number is found
    }
    }
    return max; // Return the maximum element found
    }
    public static void main(String[] args) {
    int[] numbers = {5, 2, 9, 1, 7};
    Integer maxNumber = findMaxElement(numbers);
    if (maxNumber != null) {
    System.out.println("Array: " + java.util.Arrays.toString(numbers));
    System.out.println("Maximum Element: " + maxNumber);
```

```
} else {
System.out.println("Empty array");
}
}
}
// Output:
// Array: [5, 2, 9, 1, 7]
// Maximum Element: 9
```

87. REVERSE AN ARRAY

```
i mport java.util.Arrays ;
    public class ArrayReverser {
    public static void reverseArray(int[] arr) {
    int left = 0;
    int right = arr.length - 1;
    // Swap elements until the pointers meet
    while (left < right) {
    int temp = arr[left];
    arr[left] = arr[right];
    arr[right] = temp;
    left++;
    right—;
    }
    }
    public static void main(String[] args) {
    int[] originalArray = {1, 2, 3, 4, 5};
    int[] reversedArray = originalArray.clone(); // Clone the original array
    reverseArray(reversedArray); // Reverse the cloned array
    System.out.println("Original Array: " + Arrays.toString(originalArray));
    System.out.println("Reversed Array: " +
Arrays.toString(reversedArray));
```

```
}
}
// Output:
// Original Array: [1, 2, 3, 4, 5]
// Reversed Array: [5, 4, 3, 2, 1]
```

88. CALCULATE THE POWER OF A NUMBER

```
p ublic class PowerCalculator {
    // Method to calculate power using double values
    public static double calculatePowerWithPow(double base, double exponent) {
    return Math.pow(base, exponent);
    }
    // Method to calculate power using integer base and exponent
    public static int calculatePowerWithIntegerExponent(int base, int exponent) {
    int result = 1;
    for (int i = 0; i < exponent; i++) {
    result *= base; // Multiply base exponent times
    }
    return result;
    }
    public static void main(String[] args) {
    double baseNumberDouble = 2.0;
    double exponentNumberDouble = 3.0;
    double resultWithPow = calculatePowerWithPow(baseNumberDouble, exponentNumberDouble);
```

```
    System.out.printf("%.1f to the power of %.1f using Math.pow(): %.1f%n",
    baseNumberDouble, exponentNumberDouble, resultWithPow);
    int baseNumberInt = 2;
    int exponentNumberInt = 3;
    int resultWithIntegerPow = calculatePowerWithIntegerExponent(baseNumberInt, exponentNumberInt);
    System.out.printf("%d to the power of %d using the integer method: %d%n",
    baseNumberInt, exponentNumberInt, resultWithIntegerPow);
    }
    }
    // Output:
    // 2.0 to the power of 3.0 using Math.pow(): 8.0
    // 2 to the power of 3 using the integer method: 8
```

89. FIND THE MINIMUM ELEMENT IN AN ARRAY

```
i mport java.util.Arrays ;
    public class MinElementFinder {
    // Method to find the minimum element in an array
    public static Integer findMinElement(int[] arr) {
    // Check if the array is not empty
    if (arr.length == 0) {
    return null; // Return null for empty array
    }
    // Initialize min with the first element
    int min = arr[0];
    // Loop through the array to find the minimum element
    for (int num : arr) {
    if (num < min) {
    min = num; // Update min if current number is smaller
    }
    }
    return min;
    }
    public static void main(String[] args) {
    int[] numbers = {5, 2, 9, 1, 7};
```

```
Integer minNumber = findMinElement(numbers);
if (minNumber != null) {
System.out.println("List: " + Arrays.toString(numbers));
System.out.println("Minimum Element: " + minNumber);
} else {
System.out.println("Empty list");
}
}
}
// Output:
// List: [5, 2, 9, 1, 7]
// Minimum Element: 1
```

90. CONVERT MINUTES TO HOURS AND MINUTES

```
p ublic class TimeConverter {
    // Method to convert total minutes to hours and minutes
    public static String convertMinutesToHoursAndMinutes(int totalMinutes) {
    // Calculate hours and remaining minutes
    int hours = totalMinutes / 60;
    int minutes = totalMinutes % 60;
    // Construct the result string
    return hours + " hours and " + minutes + " minutes";
    }
    public static void main(String[] args) {
    int totalMinutes = 135;
    String convertedTime = convertMinutesToHoursAndMinutes(totalMinutes);
    System.out.println(totalMinutes + " minutes is equivalent to: " + convertedTime);
    }
    }
    // Output:
    // 135 minutes is equivalent to: 2 hours and 15 minutes
```

91. FIND THE SUM OF DIGITS IN A NUMBER

```
p ublic class SumOfDigits {
    // Method to calculate the sum of digits in a number
    public static int sumOfDigits(int number) {
    int sum = 0;
    // Convert number to string and iterate over each character
    for (char digit : String.valueOf(number).toCharArray()) {
    // Convert character to digit and add to sum
    sum += Character.getNumericValue(digit);
    }
    return sum;
    }
    public static void main(String[] args) {
    int inputNumber = 12345;
    int result = sumOfDigits(inputNumber);
    System.out.println("The sum of digits in " + inputNumber + " is: " +
result);
    }
    }
    // Output:
    // The sum of digits in 12345 is: 15
```

92. LIKE VS. DISLKES

```
p ublic class LikeDislike {
    // Method to determine the final state based on button presses
    public static String likeOrDislike(String[] buttons) {
    String state = "Nothing";
    for (String button : buttons) {
    switch (state) {
    case "Nothing":
    if (button.equals("Like")) {
    state = "Like";
    } else if (button.equals("Dislike")) {
    state = "Dislike";
    }
    break;
    case "Like":
    if (button.equals("Like")) {
    state = "Nothing";
    } else if (button.equals("Dislike")) {
    state = "Dislike";
    }
    break;
    case "Dislike":
```

```
if (button.equals("Like")) {
state = "Like";
} else if (button.equals("Dislike")) {
state = "Nothing";
}
break;
}
}
return state;
}
public static void main(String[] args) {
String[][] testCases = {
{"Dislike"},
{"Like", "Like"},
{"Dislike", "Like"},
{"Like", "Dislike", "Dislike"},
};
for (String[] buttons : testCases) {
System.out.println("Final state: " + likeOrDislike(buttons));
}
}
}
// Output:
```

// Final state: Dislike

// Final state: Nothing

// Final state: Like

// Final state: Nothing

93. IS A VALID NUMBER?

```
p ublic class PhoneNumberValidator {
    // Method to check if the phone number is valid
    public static boolean isValidPhoneNumber(String phoneNumber) {
    // Check length
    if (phoneNumber.length() != 14) {
    return false;
    }
    // Check format
    for (int i = 0; i < phoneNumber.length(); i++) {
    char c = phoneNumber.charAt(i);
    switch (i) {
    case 0:
    if (c != '(') return false; // First character must be '('
    break;
    case 1: case 2: case 3:
    if (!Character.isDigit(c)) return false; // Next three characters must be
digits
    break;
    case 4:
    if (c != ')') return false; // Fifth character must be ')'
    break;
```

```
    case 5:
    if (c != ' ') return false; // Sixth character must be a space
    break;
    case 6: case 7: case 8:
    if (!Character.isDigit(c)) return false; // Next three characters must be
digits
    break;
    case 9:
    if (c != '-') return false; // Tenth character must be '-'
    break;
    case 10: case 11: case 12: case 13:
    if (!Character.isDigit(c)) return false; // Last four characters must be
digits
    break;
    default:
    return false; // Should never reach here
    }
    }
    return true; // If all checks pass, return true
    }
    public static void main(String[] args) {
    String[] testNumbers = {
    "(123) 456-7890",
```

```
"1111)555 2345",
"098) 123 4567",
};
for (String number : testNumbers) {
System.out.println("Is '" + number + "' a valid phone number? " +
isValidPhoneNumber(number));
}
}
}
// Output:
// Is '(123) 456-7890' a valid phone number? true
// Is '1111)555 2345' a valid phone number? false
// Is '098) 123 4567' a valid phone number? false
```

94. CALCULATE SIMPLE INTEREST

```
p ublic class SimpleInterestCalculator {
    // Method to calculate simple interest
    public static double calculateSimpleInterest(double principal, double rate, double time) throws IllegalArgumentException {
    // Check if the inputs are valid positive numbers
    if (principal <= 0.0 || rate <= 0.0 || time <= 0.0) {
    throw new IllegalArgumentException("Invalid inputs. Please provide valid positive numbers.");
    }
    // Calculate simple interest
    return (principal * rate * time) / 100.0;
    }
    public static void main(String[] args) {
    double principalAmount = 1000.0;
    double interestRate = 5.0;  // 5%
    double investmentTime = 2.0; // 2 years
    try {
    double interestAmount = calculateSimpleInterest(principalAmount, interestRate, investmentTime);
    System.out.printf("Principal Amount: $%.2f%n", principalAmount);
    System.out.printf("Interest Rate: %.2f%%%n", interestRate);
    System.out.printf("Investment Time: %.2f years%n", investmentTime);
```

```
System.out.printf("Simple Interest: $%.2f%n", interestAmount);
} catch (IllegalArgumentException e) {
System.out.println(e.getMessage());
}
}
}
// Output:
// Principal Amount: $1000.00
// Interest Rate: 5.00%
// Investment Time: 2.00 years
// Simple Interest: $100.00
```

95. MINI SUDOKU

```
i mport java.util.HashSet ;
    public class MiniSudoku {
    // Method to check if the 3x3 grid is a valid mini Sudoku
    public static boolean isMiniSudoku(int[][] square) {
    HashSet<Integer> seen = new HashSet<>();
    // Check if the grid is 3x3
    if (square.length != 3 || square[0].length != 3) {
    return false; // Invalid size
    }
    for (int[] row : square) {
    for (int num : row) {
    // Check if the number is within the valid range (1-9)
    if (num < 1 || num > 9) {
    return false;
    }
    // Check for duplicates
    if (!seen.add(num)) {
    return false;
    }
    }
    }
```

```
// If all numbers are seen exactly once and are in range, return true
return true;
}
public static void main(String[] args) {
int[][] validSudoku = {
{1, 3, 2},
{9, 7, 8},
{4, 5, 6}
};
int[][] invalidSudoku1 = {
{1, 1, 3},
{6, 5, 4},
{8, 7, 9}
};
int[][] invalidSudoku2 = {
{0, 1, 2},
{6, 4, 5},
{9, 8, 7}
};
int[][] validSudoku2 = {
{8, 9, 2},
{5, 6, 1},
{3, 7, 4}
```

```
    };
    System.out.println("isMiniSudoku(validSudoku) = " +
isMiniSudoku(validSudoku)); // true
    System.out.println("isMiniSudoku(invalidSudoku1) = " +
isMiniSudoku(invalidSudoku1)); // false
    System.out.println("isMiniSudoku(invalidSudoku2) = " +
isMiniSudoku(invalidSudoku2)); // false
    System.out.println("isMiniSudoku(validSudoku2) = " +
isMiniSudoku(validSudoku2)); // true
    }
    }
    // Output:
    // isMiniSudoku(validSudoku) = true
    // isMiniSudoku(invalidSudoku1) = false
    // isMiniSudoku(invalidSudoku2) = false
    // isMiniSudoku(validSudoku2) = true
```

96. CHECK IF A NUMBER IS A PERFECT NUMBER

```
p ublic class PerfectNumber {
    // Method to check if a number is a perfect number
    public static boolean isPerfectNumber(int number) {
    if (number < 2) {
    return false; // 1 and below cannot be perfect numbers
    }
    // Calculate the sum of proper divisors
    int sumOfDivisors = 1; // Start with 1, as 1 is a divisor of any number
    int sqrt = (int) Math.sqrt(number);
    for (int i = 2; i <= sqrt; i++) {
    if (number % i == 0) {
    sumOfDivisors += i;
    if (i != number / i) {
    sumOfDivisors += number / i;
    }
    }
    }
    // Check if the sum of the divisors equals the original number
    return sumOfDivisors == number;
    }
```

```
public static void main(String[] args) {
int testNumber = 28;
boolean result = isPerfectNumber(testNumber);
System.out.println("Is " + testNumber + " a perfect number? " + result);
}
}
// Output:
// Is 28 a perfect number? true
```

97. CALCULATE THE VOLUME OF A CYLINDER

```
p ublic class CylinderVolumeCalculator {
    public static double calculateCylinderVolume(double radius, double height) throws IllegalArgumentException {
    // Check if the inputs are valid positive numbers
    if (radius <= 0 || height <= 0) {
    throw new IllegalArgumentException("Invalid inputs. Please provide valid positive numbers.");
    }
    // Calculate the volume of the cylinder
    double volume = Math.PI * Math.pow(radius, 2) * height;
    return volume;
    }
    public static void main(String[] args) {
    double cylinderRadius = 5.0;
    double cylinderHeight = 10.0;
    try {
    double volume = calculateCylinderVolume(cylinderRadius, cylinderHeight);
    System.out.printf("Cylinder Volume: %.2f cubic units%n", volume);
    } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
```

```
}
}
}
/*
Output:
Cylinder Volume: 785.40 cubic units
*/
```

98. GET STUDENT TOP NOTES

```
i mport java.util.ArrayList ;
    import java.util.Collections;
    import java.util.List;
    class Student {
    private int id;
    private String name;
    private List<Integer> notes;
    public Student(int id, String name, List<Integer> notes) {
    this.id = id;
    this.name = name;
    this.notes = notes;
    }
    public int getTopNote() {
    // If there are no notes, return 0 as a default
    return notes.isEmpty() ? 0 : Collections.max(notes);
    }
    }
    public class Main {
    public static List<Integer> getStudentTopNotes(List<Student> students)
{
    List<Integer> topNotes = new ArrayList<>();
```

```
for (Student student : students) {
topNotes.add(student.getTopNote());
}
return topNotes;
}
public static void main(String[] args) {
List<Student> students = new ArrayList<>();
students.add(new Student(1, "Jacek", List.of(5, 3, 4, 2, 5, 5)));
students.add(new Student(2, "Ewa", List.of(2, 3, 3, 3, 2, 5)));
students.add(new Student(3, "Zygmunt", List.of(2, 2, 4, 4, 3, 3)));
List<Integer> topNotes = getStudentTopNotes(students);
System.out.println("Top notes: " + topNotes);
}
}
// Output: Top notes: [5, 5, 4]
```

99. FIND THE INTERSECTION OF TWO ARRAYS

```
i mport java.util.ArrayList ;
    import java.util.HashSet;
    import java.util.List;
    public class Main {
    public static List<Integer> findIntersection(int[] arr1, int[] arr2) {
    // Convert the first array into a HashSet for efficient membership testing
    HashSet<Integer> set1 = new HashSet<>();
    for (int num : arr1) {
    set1.add(num);
    }
    // Use a List to collect the intersection
    List<Integer> intersection = new ArrayList<>();
    for (int num : arr2) {
    if (set1.contains(num)) {
    intersection.add(num);
    }
    }
    return intersection;
    }
    public static void main(String[] args) {
    int[] arr1 = {1, 2, 3, 4, 5};
```

```
int[] arr2 = {3, 4, 5, 6, 7};
List<Integer> intersection = findIntersection(arr1, arr2);
System.out.println("Intersection of Arrays: " + intersection);
}
}
// Output: Intersection of Arrays: [3, 4, 5]
```

100. CONVERT FEET TO METERS

```
p ublic class Main {
    public static double feetToMeters(double feet) {
    // Converts feet to meters
    return feet * 0.3048;
    }
    public static void main(String[] args) {
    double feetValue = 10.0;
    double metersValue = feetToMeters(feetValue);
    System.out.printf("%.2f feet is equal to %.2f meters%n", feetValue,
metersValue);
    }
    }
    // Output: 10.00 feet is equal to 3.05 meters
```

101. CONVERT DAYS TO YEARS, MONTHS, AND DAYS

```
p ublic class Main {
    public static int[] convertDaysToYearsMonthsDays(int days) {
    // Calculate years
    int years = days / 365;
    // Calculate remaining days after years
    int remainingDaysAfterYears = days % 365;
    // Calculate months
    int months = remainingDaysAfterYears / 30;
    // Calculate remaining days after months
    int remainingDaysAfterMonths = remainingDaysAfterYears % 30;
    return new int[] { years, months, remainingDaysAfterMonths };
    }
    public static void main(String[] args) {
    int totalDays = 1000;
    int[] result = convertDaysToYearsMonthsDays(totalDays);
    System.out.printf("%d days is approximately %d years, %d months, and
%d days.%n", totalDays, result[0], result[1], result[2]);
    }
    }
    // Output: 1000 days is approximately 2 years, 9 months, and 0 days.
```

102. FIND THE MEDIAN OF AN ARRAY

```
i mport java.util.Arrays ;
    public class Main {
    public static double findMedian(int[] arr) {
    // Check if the list is valid
    if (arr.length == 0) {
    throw new IllegalArgumentException("Invalid input. Please provide a non-empty list.");
    }
    // Sort the array
    Arrays.sort(arr);
    int len = arr.length;
    int middleIndex = len / 2;
    // Calculate the median
    if (len % 2 == 0) {
    // If the list has an even number of elements, return the average of the two middle elements
    return (arr[middleIndex - 1] + arr[middleIndex]) / 2.0;
    } else {
    // If the list has an odd number of elements, return the middle element
    return arr[middleIndex];
    }
```

```
}
public static void main(String[] args) {
int[] numbers = {5, 2, 8, 1, 7, 3};
double median = findMedian(numbers);
System.out.printf("Median: %.1f%n", median);
}
}
// Output: Median: 4.0
```

103. CALCULATE THE DISTANCE BETWEEN TWO POINTS

```
p ublic class Main {
    public static double calculateDistance(double x1, double y1, double x2, double y2) {
    // Calculate the distance using the distance formula
    return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
    }
    public static void main(String[] args) {
    double x1 = 1.0;
    double y1 = 2.0;
    double x2 = 4.0;
    double y2 = 6.0;
    double result = calculateDistance(x1, y1, x2, y2);
    System.out.printf("The distance between (%.1f, %.1f) and (%.1f, %.1f) is %.2f%n", x1, y1, x2, y2, result);
    }
    }
    // Output: The distance between (1.0, 2.0) and (4.0, 6.0) is 5.00
```

104. CHECK IF A NUMBER IS A PERFECT SQUARE

```
p ublic class Main {
    public static boolean isPerfectSquare(double number) {
    // Check if the number is a non-negative value
    if (number < 0) {
    return false;
    }
    // Calculate the square root of the number
    double squareRoot = Math.sqrt(number);
    // Check if the square of the integer part of the square root is equal to the original number
    return squareRoot == Math.floor(squareRoot);
    }
    public static void main(String[] args) {
    double testNumber = 25.0;
    boolean result = isPerfectSquare(testNumber);
    System.out.printf("Is %.1f a perfect square? %b%n", testNumber, result);
    }
    }
    // Output: Is 25.0 a perfect square? true
```

105. FIND THE AREA OF A RECTANGLE

```
p ublic class Main {
    public static double calculateRectangleArea(double length, double width) {
    // Check if the inputs are valid positive numbers
    if (length <= 0.0 || width <= 0.0) {
    System.out.println("Invalid inputs. Please provide valid positive numbers for length and width.");
    return 0.0;
    }
    // Calculate the area of the rectangle
    return length * width;
    }
    public static void main(String[] args) {
    double rectangleLength = 5.0;
    double rectangleWidth = 8.0;
    double result = calculateRectangleArea(rectangleLength, rectangleWidth);
    System.out.printf("The area of the rectangle with length %.1f and width %.1f is %.2f%n",
    rectangleLength, rectangleWidth, result);
    }
```

```
}
// Output: The area of the rectangle with length 5.0 and width 8.0 is 40.00
```

106. CONVERT BINARY TO DECIMAL

```
p ublic class Main {
    public static int binaryToDecimal(String binaryString) throws
IllegalArgumentException {
    // Check if the input string contains only '0' and '1'
    if (!binaryString.matches("[01]+")) {
    throw new IllegalArgumentException("Invalid input. Please provide a
valid binary string.");
    }
    // Convert binary string to decimal
    return Integer.parseInt(binaryString, 2);
    }
    public static void main(String[] args) {
    String binaryNumber = "1101";
    try {
    int decimalResult = binaryToDecimal(binaryNumber);
    System.out.printf("The decimal equivalent of binary %s is %d%n",
binaryNumber, decimalResult);
    } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
    }
    }
    }
```

```
// Output: The decimal equivalent of binary 1101 is 13
```

107. COUNT THE NUMBER OF WORDS IN A SENTENCE

```
p ublic class Main {
    public static int countWords(String sentence) throws
IllegalArgumentException {
    // Check if the input is a valid non-empty string
    if (sentence == null || sentence.trim().isEmpty()) {
    throw new IllegalArgumentException("Invalid input. Please provide a
valid sentence.");
    }
    // Count the words in the sentence
    String[] words = sentence.trim().split("\\s+");
    return words.length;
    }
    public static void main(String[] args) {
    String sentence = "This is a sample sentence.";
    try {
    int wordCount = countWords(sentence);
    System.out.printf("The sentence \"%s\" has %d words.%n", sentence,
wordCount);
    } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
    }
```

```
}
}
// Output: The sentence "This is a sample sentence." has 5 words.
```

108. FIND THE UNION OF TWO ARRAYS

```
i mport java.util.Arrays ;
    import java.util.HashSet;
    public class Main {
    public static <T> HashSet<T> findUnion(T[] arr1, T[] arr2) {
    // Create a HashSet to store the union of both arrays
    HashSet<T> unionSet = new HashSet<>();
    // Insert elements of both arrays into the HashSet
    unionSet.addAll(Arrays.asList(arr1));
    unionSet.addAll(Arrays.asList(arr2));
    return unionSet;
    }
    public static void main(String[] args) {
    Integer[] array1 = {1, 2, 3, 4, 5};
    Integer[] array2 = {3, 4, 5, 6, 7};
    // Get the union of both arrays
    HashSet<Integer> unionResult = findUnion(array1, array2);
    // Display the result
    System.out.println("Union of Arrays: " + unionResult);
    }
    }
```

```
// Output: Union of Arrays: [1, 2, 3, 4, 5, 6, 7]
```

109. SCORING SYSTEM

```
i mport java.util.HashMap ;
    public class ScoringSystem {
    public static int[] calculateScores(String scoreString) {
    // Initialize the scores for Andy (A), Ben (B), and Charlotte (C)
    HashMap<Character, Integer> scores = new HashMap<>();
    scores.put('A', 0);
    scores.put('B', 0);
    scores.put('C', 0);
    // Count occurrences of each letter in the string
    for (char ch : scoreString.toCharArray()) {
    scores.put(ch, scores.getOrDefault(ch, 0) + 1);
    }
    // Return the scores in the order: Andy (A), Ben (B), Charlotte (C)
    return new int[] {
    scores.getOrDefault('A', 0),
    scores.getOrDefault('B', 0),
    scores.getOrDefault('C', 0)
    };
    }
    public static void main(String[] args) {
    // Example usage
```

```
    System.out.println(java.util.Arrays.toString(calculateScores("A")));
    System.out.println(java.util.Arrays.toString(calculateScores("ABC")));
    System.out.println(java.util.Arrays.toString(calculateScores("ABCBACC")));
    }
    }
    /*
    Output:
    [1, 0, 0]
    [1, 1, 1]
    [2, 2, 3]
    */
```

110. CHECK IF A NUMBER IS A STRONG NUMBER

```
p ublic class StrongNumberChecker {
    // Method to calculate factorial of a number
    public static int calculateFactorial(int n) {
    int result = 1;
    for (int i = 2; i <= n; i++) {
    result *= i;
    }
    return result;
    }
    // Method to check if a number is a strong number
    public static boolean isStrongNumber(int num) {
    int originalNumber = num;
    int digitFactorialSum = 0;
    // Calculate the sum of the factorial of each digit
    while (num > 0) {
    int digit = num % 10; // Get the last digit
    digitFactorialSum += calculateFactorial(digit);
    num /= 10; // Remove the last digit
    }
    // Check if the sum equals the original number
```

```
return digitFactorialSum == originalNumber;
}
public static void main(String[] args) {
int testNumber = 145;
boolean result = isStrongNumber(testNumber);
System.out.println(testNumber + " is a strong number: " + result);
}
}
/*
Output:
145 is a strong number: true
*/
```

111. CHECK IF A NUMBER IS A NARCISSISTIC NUMBER

```
p ublic class NarcissisticNumberChecker {
    // Method to check if a number is a narcissistic number
    public static boolean isNarcissisticNumber(int num) {
    // Convert the number to a string
    String numStr = Integer.toString(num);
    int numDigits = numStr.length();
    // Calculate the sum of each digit raised to the power of the number of digits
    int total = 0;
    for (char digitChar : numStr.toCharArray()) {
    int digit = Character.getNumericValue(digitChar);
    total += Math.pow(digit, numDigits);
    }
    // Check if the total equals the original number
    return total == num;
    }
    public static void main(String[] args) {
    int testNumber = 1634;
    boolean result = isNarcissisticNumber(testNumber);
    System.out.println(testNumber + " is a Narcissistic Number: " + result);
```

```
}
}
/*
Output:
1634 is a Narcissistic Number: true
*/
```

112. COUNT THE NUMBER OF CONSONANTS IN A STRING

```
p ublic class ConsonantCounter {
    // Method to count the number of consonants in a string
    public static int countConsonants(String inputStr) {
    // Define a string of consonant characters
    String consonants = "bcdfghjklmnpqrstvwxyz";
    int count = 0;
    // Convert the input string to lowercase and iterate through each
character
    for (char c : inputStr.toLowerCase().toCharArray()) {
    // Check if the character is a consonant
    if (consonants.indexOf(c) != -1) {
    count++;
    }
    }
    return count;
    }
    public static void main(String[] args) {
    String testString = "Hello World";
    int result = countConsonants(testString);
    System.out.println("The number of consonants in '" + testString + "' is: "
+ result);
```

```
}
}
/*
Output:
The number of consonants in 'Hello World' is: 7
*/
```

113. CHECK IF A NUMBER IS A TRIANGULAR NUMBER

```
p ublic class TriangularNumberChecker {
    // Method to check if a number is a triangular number
    public static boolean isTriangularNumber(int num) {
    // Check if the input is a non-negative integer
    if (num <= 0) {
    return false; // Zero and negative numbers are not considered triangular numbers
    }
    int total = 0;
    int n = 1;
    // Iterate through natural numbers until the sum exceeds or equals the input number
    while (total < num) {
    total += n;
    n++;
    }
    // Check if the input number is equal to a triangular number
    return total == num;
    }
    public static void main(String[] args) {
    int testNumber = 10;
```

```
boolean result = isTriangularNumber(testNumber);
System.out.println(testNumber + " is a triangular number: " + result);
}
}
/*
Output:
10 is a triangular number: true
*/
```

114. FIND THE AREA OF A TRAPEZOID

```
p ublic class TrapezoidAreaCalculator {
    // Method to calculate the area of a trapezoid
    public static double trapezoidArea(double base1, double base2, double height) throws IllegalArgumentException {
    // Check if the inputs are valid numbers
    if (base1 <= 0 || base2 <= 0 || height <= 0) {
    throw new IllegalArgumentException("Invalid input. Please provide valid positive numbers.");
    }
    // Calculate the area of the trapezoid
    return 0.5 * height * (base1 + base2);
    }
    public static void main(String[] args) {
    double base1Length = 5.0;
    double base2Length = 9.0;
    double trapezoidHeight = 4.0;
    try {
    double area = trapezoidArea(base1Length, base2Length, trapezoidHeight);
    System.out.printf("The area of the trapezoid is: %.2f%n", area);
    } catch (IllegalArgumentException e) {
```

```
System.out.println(e.getMessage());
}
}
}
/*
Output:
The area of the trapezoid is: 28.00
*/
```

115. ABBREVIATIONS UNIQUE?

```
i mport java.util.HashMap ;
    import java.util.List;
    public class Main {
    public static boolean uniqueAbbrev(String[] abbreviations, String[]
words) {
    // Create a map where each abbreviation maps to the words it can
represent
    HashMap<String, List<String>> abbrevMap = new HashMap<>();
    for (String word : words) {
    String abbrev = word.substring(0, 1); // Use the first character as
abbreviation
    abbrevMap.computeIfAbsent(abbrev, k -> new java.util.ArrayList<>
()).add(word);
    }
    // Check that each abbreviation is unique
    for (String abbrev : abbreviations) {
    List<String> correspondingWords = abbrevMap.get(abbrev);
    if (correspondingWords == null || correspondingWords.size() != 1) {
    return false; // Abbreviation is not unique
    }
    }
    return true; // All abbreviations are unique
```

```
    }
    public static void main(String[] args) {
    // Example usage
    String[] abbreviations1 = {"ho", "h", "ha"};
    String[] words1 = {"house", "hope", "happy"};
    System.out.println("uniqueAbbrev([\"ho\", \"h\", \"ha\"], [\"house\", \"hope\", \"happy\"]) ➞ " + uniqueAbbrev(abbreviations1, words1));
    String[] abbreviations2 = {"s", "t", "v"};
    String[] words2 = {"stamina", "television", "vindaloo"};
    System.out.println("uniqueAbbrev([\"s\", \"t\", \"v\"], [\"stamina\", \"television\", \"vindaloo\"]) ➞ " + uniqueAbbrev(abbreviations2, words2));
    String[] abbreviations3 = {"bi", "ba", "bat"};
    String[] words3 = {"big", "bard", "battery"};
    System.out.println("uniqueAbbrev([\"bi\", \"ba\", \"bat\"], [\"big\", \"bard\", \"battery\"]) ➞ " + uniqueAbbrev(abbreviations3, words3));
    String[] abbreviations4 = {"mo", "ma", "me"};
    String[] words4 = {"moment", "many", "mean"};
    System.out.println("uniqueAbbrev([\"mo\", \"ma\", \"me\"], [\"moment\", \"many\", \"mean\"]) ➞ " + uniqueAbbrev(abbreviations4, words4));
    }
    }
    // Output:
```

// uniqueAbbrev(["ho", "h", "ha"], ["house", "hope", "happy"]) ➞ true

// uniqueAbbrev(["s", "t", "v"], ["stamina", "television", "vindaloo"]) ➞ false

// uniqueAbbrev(["bi", "ba", "bat"], ["big", "bard", "battery"]) ➞ false

// uniqueAbbrev(["mo", "ma", "me"], ["moment", "many", "mean"]) ➞ true

116. CHECK IF A NUMBER IS A FIBONACCI NUMBER

```
p ublic class Main {
    // Helper function to check if a number is a perfect square
    private static boolean isPerfectSquare(long n) {
    long sqrt = (long) Math.sqrt(n);
    return sqrt * sqrt == n;
    }
    public static boolean isFibonacciNumber(long num) {
    // A number is a Fibonacci number if and only if one of (5 * num^2 + 4) or (5 * num^2 - 4) is a perfect square
    long numSquared = num * num;
    return isPerfectSquare(5 * numSquared + 4) || isPerfectSquare(5 * numSquared - 4);
    }
    public static void main(String[] args) {
    long testNumber = 8;
    boolean result = isFibonacciNumber(testNumber);
    System.out.println(testNumber + " is a Fibonacci number: " + result);
    }
    }
    // Output:
    // 8 is a Fibonacci number: true
```

117. FIND THE PERIMETER OF A RECTANGLE

```
p ublic class Main {
    public static double rectanglePerimeter(double length, double width)
throws IllegalArgumentException {
    // Check if the inputs are valid numbers
    if (length <= 0.0 || width <= 0.0) {
    throw new IllegalArgumentException("Invalid input. Please provide
valid positive numbers.");
    }
    // Calculate the perimeter of the rectangle
    return 2.0 * (length + width);
    }
    public static void main(String[] args) {
    double length = 5.0;
    double width = 8.0;
    try {
    double perimeter = rectanglePerimeter(length, width);
    System.out.println("The perimeter of the rectangle is: " + perimeter);
    } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
    }
    }
```

```
}
// Output:
// The perimeter of the rectangle is: 26.0
```

118. CLUB ENTRY

```
p ublic class Main {
    public static Integer clubEntry(String word) {
    // Find the doubled letter
    Character doubledLetter = null;
    for (int i = 0; i < word.length() - 1; i++) {
    if (word.charAt(i) == word.charAt(i + 1)) {
    doubledLetter = word.charAt(i);
    break;
    }
    }
    // Ensure that a doubled letter was found
    if (doubledLetter == null) {
    return null; // No doubled letter found
    }
    // Calculate the position of the letter in the alphabet
    int position = doubledLetter - 'a' + 1;
    // Multiply the position by 4
    int result = position * 4;
    return result;
    }
    public static void main(String[] args) {
```

```
    String[] words = {"hill", "apple", "bee"};
    for (String word : words) {
    Integer number = clubEntry(word);
    if (number != null) {
    System.out.println("clubEntry(\"" + word + "\") ➞ " + number);
    } else {
    System.out.println("clubEntry(\"" + word + "\") ➞ No doubled letter found");
    }
    }
    }
    }
    // Output:
    // clubEntry("hill") ➞ 32
    // clubEntry("apple") ➞ No doubled letter found
    // clubEntry("bee") ➞ 8
```

119. CHECK IF A STRING IS ANAGRAM OF ANOTHER STRING

```
i mport java.util.HashMap ;
    public class Main {
    public static boolean areAnagrams(String str1, String str2) {
    // Function to clean and count characters
    HashMap<Character, Integer> cleanAndCount(String s) {
    HashMap<Character, Integer> count = new HashMap<>();
    for (char c : s.toCharArray()) {
    if (Character.isAlphabetic(c)) {
    char lowerChar = Character.toLowerCase(c);
    count.put(lowerChar, count.getOrDefault(lowerChar, 0) + 1);
    }
    }
    return count;
    }
    // Get character counts for both strings
    HashMap<Character, Integer> count1 = cleanAndCount(str1);
    HashMap<Character, Integer> count2 = cleanAndCount(str2);
    // Compare the two frequency counts
    return count1.equals(count2);
    }
```

```
    public static void main(String[] args) {
    String string1 = "listen";
    String string2 = "silent";
    boolean result = areAnagrams(string1, string2);
    System.out.println(string1 + " and " + string2 + " are anagrams: " +
result);
    }
    }
    // Output:
    // listen and silent are anagrams: true
```

120. GENERATE PASCAL'S TRIANGLE

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class Main {
    public static List<List<Integer>> generatePascalsTriangle(int
numRows) {
    List<List<Integer>> triangle = new ArrayList<>();
    if (numRows == 0) {
    return triangle;
    }
    // Initialize the first row
    triangle.add(new ArrayList<>());
    triangle.get(0).add(1);
    for (int i = 1; i < numRows; i++) {
    List<Integer> row = new ArrayList<>();
    row.add(1); // First element of each row is 1
    // Compute the values based on the previous row
    List<Integer> prevRow = triangle.get(i - 1);
    for (int j = 1; j < i; j++) {
    row.add(prevRow.get(j - 1) + prevRow.get(j));
    }
    row.add(1); // Last element of each row is 1
```

```
    triangle.add(row);
    }
    return triangle;
    }
    public static void main(String[] args) {
    int numberOfRows = 5;
    List<List<Integer>> triangle =
generatePascalsTriangle(numberOfRows);
    System.out.println("Pascal's Triangle with " + numberOfRows + "
rows:");
    for (List<Integer> row : triangle) {
    System.out.println(row);
    }
    }
    }
    // Output:
    // Pascal's Triangle with 5 rows:
    // [1]
    // [1, 1]
    // [1, 2, 1]
    // [1, 3, 3, 1]
    // [1, 4, 6, 4, 1]
```

121. CONVERT DECIMAL TO ROMAN NUMERALS

```
p ublic class Main {
    public static String decimalToRoman(int num) throws
IllegalArgumentException {
    if (num <= 0 || num > 3999) {
    throw new IllegalArgumentException("Invalid input. Please provide a
valid positive integer within the range 1 to 3999.");
    }
    // Define the Roman numeral symbols and their values
    String[] romanSymbols = {
    "M", "CM", "D", "CD",
    "C", "XC", "L", "XL",
    "X", "IX", "V", "IV", "I"
    };
    int[] values = {
    1000, 900, 500, 400,
    100, 90, 50, 40,
    10, 9, 5, 4, 1
    };
    StringBuilder result = new StringBuilder();
    // Convert decimal to Roman numeral
    for (int i = 0; i < values.length; i++) {
```

```
    while (num >= values[i]) {
    result.append(romanSymbols[i]);
    num -= values[i];
    }
    }
    return result.toString();
    }
    public static void main(String[] args) {
    int decimalNumber = 1984;
    try {
    String romanNumeral = decimalToRoman(decimalNumber);
    System.out.println("The Roman numeral representation of " +
decimalNumber + " is: " + romanNumeral);
    } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
    }
    }
    }
    // Output:
    // The Roman numeral representation of 1984 is: MCMLXXXIV
    }
    // The Roman numeral representation of 1984 is: MCMLXXXIV
```

122. FIND THE AREA OF A PARALLELOGRAM

```
p ublic class Main {
    public static double parallelogramArea(double base, double height)
throws IllegalArgumentException {
    if (base > 0.0 && height > 0.0) {
    return base * height;
    } else {
    throw new IllegalArgumentException("Invalid input. Please provide
valid positive numbers.");
    }
    }
    public static void main(String[] args) {
    double base = 6.0;
    double height = 8.0;
    try {
    double area = parallelogramArea(base, height);
    System.out.println("The area of the parallelogram is: " + area);
    } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
    }
    }
    }
```

```
// Output:
// The area of the parallelogram is: 48.0
```

123. SUPERHEROES

```
i mport java.util.ArrayList ;
    import java.util.Collections;
    import java.util.List;
    public class Main {
    public static List<String> superheroes(String[] names) {
    List<String> filtered = new ArrayList<>();
    // Filter names ending with "man"
    for (String name : names) {
    if (name.toLowerCase().endsWith("man")) {
    filtered.add(name);
    }
    }
    // Sort the filtered list
    Collections.sort(filtered);
    return filtered;
    }
    public static void main(String[] args) {
    String[] heroes1 = {"Batman", "Superman", "Spider-man", "Hulk",
"Wolverine", "Wonder-Woman"};
    List<String> result1 = superheroes(heroes1);
    System.out.println(result1); // Output: [Batman, Spider-man, Superman]
```

```
    String[] heroes2 = {"Catwoman", "Deadpool", "Dr.Strange", "Captain-America", "Aquaman", "Hawkeye"};
    List<String> result2 = superheroes(heroes2);
    System.out.println(result2); // Output: [Aquaman]
    String[] heroes3 = {"Wonder-Woman", "Catwoman", "Invisible-Woman"};
    List<String> result3 = superheroes(heroes3);
    System.out.println(result3); // Output: []
    }
    }
    // Expected Output:
    // [Batman, Spider-man, Superman]
    // [Aquaman]
    // []
```

124. APPLYING DISCOUNTS

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class Main {
    public static List<Double> getDiscounts(double[] prices, String
discount) {
    // Extract the discount percentage as a double
    double discountPercentage =
Double.parseDouble(discount.replace("%", "").trim());
    double discountFactor = discountPercentage / 100.0;
    // Calculate discounted prices
    List<Double> discountedPrices = new ArrayList<>();
    for (double price : prices) {
    discountedPrices.add(price * (1.0 - discountFactor));
    }
    return discountedPrices;
    }
    public static void main(String[] args) {
    // Example usage
    double[] prices1 = {2.0, 4.0, 6.0, 11.0};
    String discount1 = "50%";
    List<Double> discountedPrices1 = getDiscounts(prices1, discount1);
```

```
System.out.println(discountedPrices1); // Output: [1.0, 2.0, 3.0, 5.5]
double[] prices2 = {10.0, 20.0, 40.0, 80.0};
String discount2 = "75%";
List<Double> discountedPrices2 = getDiscounts(prices2, discount2);
System.out.println(discountedPrices2); // Output: [7.5, 15.0, 30.0, 60.0]
double[] prices3 = {100.0};
String discount3 = "45%";
List<Double> discountedPrices3 = getDiscounts(prices3, discount3);
System.out.println(discountedPrices3); // Output: [55.0]
}
}
// Expected Output:
// [1.0, 2.0, 3.0, 5.5]
// [7.5, 15.0, 30.0, 60.0]
// [55.0]
```

125. CHECK IF A NUMBER IS A SMITH NUMBER

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class Main {
    public static boolean isPrime(long num) {
    if (num < 2) return false;
    for (long i = 2; i * i <= num; i++) {
    if (num % i == 0) return false;
    }
    return true;
    }
    public static long sumOfDigits(long num) {
    long sum = 0;
    while (num > 0) {
    sum += num % 10;
    num /= 10;
    }
    return sum;
    }
    public static List<Long> primeFactors(long num) {
    List<Long> factors = new ArrayList<>();
```

```
    for (long i = 2; i * i <= num; i++) {
    while (num % i == 0) {
    factors.add(i);
    num /= i;
    }
    }
    if (num > 1) {
    factors.add(num);
    }
    return factors;
    }
    public static boolean isSmithNumber(long num) {
    long originalSum = sumOfDigits(num);
    long primeFactorSum =
primeFactors(num).stream().mapToLong(Main::sumOfDigits).sum();
    return originalSum != primeFactorSum && !isPrime(num);
    }
    public static void main(String[] args) {
    long number = 728;
    System.out.println("Is " + number + " a Smith number? " +
isSmithNumber(number));
    }
    }
```

```
// Output: Is 728 a Smith number? false
```

126. BASIC CHESSBOARD

```
p ublic class Main {
    public static String generateChessboard() {
    int size = 8; // Size of the chessboard (8x8)
    StringBuilder chessboard = new StringBuilder();
    for (int row = 0; row < size; row++) {
    StringBuilder currentRow = new StringBuilder();
    for (int col = 0; col < size; col++) {
    // Use 'X' for black squares and ' ' for white squares
    char square = (row + col) % 2 == 1 ? 'X' : ' ';
    currentRow.append(square).append(' '); // Adding a space between
squares
    }
    // Add the current row to the chessboard and add a newline
    chessboard.append(currentRow.toString().trim()).append('\n'); // Trim
trailing space
    }
    return chessboard.toString();
    }
    public static void main(String[] args) {
    String chessboard = generateChessboard();
    System.out.print(chessboard);
```

```
}
}
// Output:
//  X  X  X  X
// X  X  X  X
//  X  X  X  X
// X  X  X  X
//  X  X  X  X
// X  X  X  X
//  X  X  X  X
// X  X  X  X
```

127. WHICH NUMBER IS NOT LIKE THE OTHERS

```
i mport java.util.HashMap ;
    import java.util.Map;
    public class Main {
    public static int unique(int[] numbers) {
    Map<Integer, Integer> counts = new HashMap<>();
    // Count occurrences of each number
    for (int num : numbers) {
    counts.put(num, counts.getOrDefault(num, 0) + 1);
    }
    // Find the unique number
    for (Map.Entry<Integer, Integer> entry : counts.entrySet()) {
    if (entry.getValue() == 1) {
    return entry.getKey();
    }
    }
    // If no unique number is found, returning 0 (as a fallback)
    return 0; // You may want to handle this case differently
    }
    public static void main(String[] args) {
    int[] numbers1 = {3, 3, 3, 7, 3, 3};
```

```
    int[] numbers2 = {0, 0, 77, 0, 0};
    int[] numbers3 = {0, 1, 1, 1, 1, 1, 1, 1};
    System.out.println("Unique number in array 1: " + unique(numbers1));
// Output: 7
    System.out.println("Unique number in array 2: " + unique(numbers2));
// Output: 77
    System.out.println("Unique number in array 3: " + unique(numbers3));
// Output: 0
    }
    }
```

128. FIND THE DISCOUNT

```
p ublic class Main {
    public static double findDiscount(double originalPrice, int
discountPercentage) {
    double discount = originalPrice * (discountPercentage / 100.0);
    return originalPrice - discount;
    }
    public static void main(String[] args) {
    // Example usage
    System.out.printf("%.2f%n", findDiscount(1500.0, 50)); // Output:
750.00
    System.out.printf("%.2f%n", findDiscount(89.0, 20));   // Output: 71.20
    System.out.printf("%.2f%n", findDiscount(100.0, 75));  // Output: 25.00
    }
    }
```

129. CHECK IF A STRING IS PANGRAM OR NOT

```
p ublic class Main {
    public static boolean isPangram(String inputStr) {
    String alphabet = "abcdefghijklmnopqrstuvwxyz";
    String lowercasedStr = inputStr.toLowerCase();
    for (char ch : alphabet.toCharArray()) {
    if (lowercasedStr.indexOf(ch) == -1) {
    return false;
    }
    }
    return true;
    }
    public static void main(String[] args) {
    // Example usage
    String inputString = "The quick brown fox jumps over the lazy dog";
    if (isPangram(inputString)) {
    System.out.println("The given string is a pangram! 🎉");
    } else {
    System.out.println("The given string is not a pangram. 😢");
    }
    }
```

}

130. COAXIAL CABLE IMPEDANCE

```
p ublic class Main {
    public static double impedanceCalculator(double dD, double dC, double
eR) {
    // Calculate the impedance using the formula
    double logTerm = Math.log10(dD / dC);
    double impedance = 60.0 / Math.sqrt(eR) * logTerm;
    return impedance;
    }
    public static void main(String[] args) {
    // Example usage
    System.out.printf("%.1f%n", impedanceCalculator(20.7, 2.0, 4.0));
    System.out.printf("%.1f%n", impedanceCalculator(5.3, 1.2, 2.2));
    System.out.printf("%.1f%n", impedanceCalculator(4.48, 1.33, 2.2));
    }
    }
    // 30.4
    // 26.1
    // 21.3
```

131. CENSOR WORDS LONGER THAN FOUR CHARACTERS

```
p ublic class Main {
    public static String censor(String text) {
    // Split the text into words
    String[] words = text.split("\\s+");
    // Transform each word, replacing those longer than four characters
    StringBuilder censoredText = new StringBuilder();
    for (String word : words) {
    if (word.length() > 4) {
    // Replace with asterisks
    String censoredWord = "*".repeat(word.length());
    censoredText.append(censoredWord).append(" ");
    } else {
    censoredText.append(word).append(" ");
    }
    }
    // Return the censored words as a single string, trimming any trailing
space
    return censoredText.toString().trim();
    }
    public static void main(String[] args) {
```

```
// Example usage
System.out.println(censor("The code is fourty"));
System.out.println(censor("Two plus three is five"));
System.out.println(censor("aaaa aaaaa 1234 12345"));
}
}
// The code is ******
// Two plus ***** is five
// aaaa ***** 1234 *****
```

132. FIND THE AREA OF AN ELLIPSE

```
p ublic class Main {
    // Method to calculate the area of an ellipse
    public static double calculateEllipseArea(double semiMajorAxis,
double semiMinorAxis) {
    return Math.PI * semiMajorAxis * semiMinorAxis;
    }
    public static void main(String[] args) {
    double semiMajorAxis = 5.0; // Replace with the semi-major axis length
    double semiMinorAxis = 3.0; // Replace with the semi-minor axis
length
    double ellipseArea = calculateEllipseArea(semiMajorAxis,
semiMinorAxis);
    System.out.printf("The area of the ellipse is: %.2f%n", ellipseArea);
    }
    }
    // The area of the ellipse is: 47.12
```

133. CHECK IF A NUMBER IS A PALINDROME IN BINARY

```
p ublic class Main {
    // Method to check if a number is a palindrome in binary
    public static boolean isBinaryPalindrome(int number) {
    String binaryRepresentation = Integer.toBinaryString(number); // Convert to binary representation
    String reversedBinary = new StringBuilder(binaryRepresentation).reverse().toString(); // Reverse the binary representation
    return binaryRepresentation.equals(reversedBinary); // Check if it is a palindrome
    }
    public static void main(String[] args) {
    int numberToCheck = 9; // Replace with the number you want to check
    if (isBinaryPalindrome(numberToCheck)) {
    System.out.printf("%d is a binary palindrome! 🎉%n", numberToCheck);
    } else {
    System.out.printf("%d is not a binary palindrome. 😢%n", numberToCheck);
    }
    }
```

```
}
// 9 is a binary palindrome! 🎉
```

134. FIND THE AREA OF A RHOMBUS

```
p ublic class Main {
    // Method to calculate the area of a rhombus given its diagonals
    public static double calculateRhombusArea(double diagonal1, double diagonal2) {
    return (diagonal1 * diagonal2) / 2.0; // Area formula for rhombus
    }
    public static void main(String[] args) {
    double diagonal1Length = 8.0; // Length of the first diagonal
    double diagonal2Length = 6.0; // Length of the second diagonal
    double rhombusArea = calculateRhombusArea(diagonal1Length, diagonal2Length);
    System.out.printf("The area of the rhombus is: %.2f%n", rhombusArea);
    }
    }
    // The area of the rhombus is: 24.00
```

135. CHECK IF A NUMBER IS A CATALAN NUMBER

```
p ublic class Main {
    // Method to calculate the binomial coefficient C(n, k)
    public static long binomialCoefficient(long n, long k) {
    long result = 1;
    if (k > n - k) {
    k = n - k; // Take advantage of symmetry
    }
    for (long i = 0; i < k; i++) {
    result *= n;
    result /= (i + 1);
    n—;
    }
    return result;
    }
    // Method to check if a number is a Catalan number
    public static boolean isCatalanNumber(long num) {
    long i = 0;
    while (true) {
    long catalan = binomialCoefficient(2 * i, i) / (i + 1);
    if (catalan == num) {
```

```
    return true; // Found a match
    } else if (catalan > num) {
    return false; // Exceeded the number
    }
    i++;
    }
    }
    public static void main(String[] args) {
    long numberToCheck = 42; // Replace with the number you want to check
    if (isCatalanNumber(numberToCheck)) {
    System.out.printf("%d is a Catalan number! 🎉%n", numberToCheck);
    } else {
    System.out.printf("%d is not a Catalan number. 😢%n", numberToCheck);
    }
    }
    }
    // 42 is a Catalan number! 🎉
```

136. FIND THE LUHN ALGORITHM CHECK DIGIT

```
p ublic class Main {
    // Method to calculate the Luhn check digit
    public static int calculateLuhnCheckDigit(long input) {
    String inputStr = Long.toString(input);
    int totalSum = 0;
    boolean isEvenPosition = inputStr.length() % 2 == 0;
    // Iterate over the digits
    for (int i = 0; i < inputStr.length(); i++) {
    int digit = Character.getNumericValue(inputStr.charAt(i));
    // Check if we are in an even position
    if (isEvenPosition) {
    // For even position, we double the digit
    if (i % 2 == 0) {
    digit *= 2;
    if (digit > 9) {
    digit -= 9; // Subtract 9 if the result is greater than 9
    }
    }
    } else {
    // For odd position, we double the digit
```

```
if (i % 2 != 0) {
digit *= 2;
if (digit > 9) {
digit -= 9; // Subtract 9 if the result is greater than 9
}
}
}
totalSum += digit; // Add the processed digit to the total sum
}
// Calculate check digit
return (10 - (totalSum % 10)) % 10;
}
public static void main(String[] args) {
long partialNumber = 123456789; // Replace with the partial number
int checkDigit = calculateLuhnCheckDigit(partialNumber);
String fullNumber = Long.toString(partialNumber) + checkDigit;
System.out.println("Partial Number: " + partialNumber);
System.out.println("Check Digit: " + checkDigit);
System.out.println("Full Number with Check Digit: " + fullNumber);
}
}
// Partial Number: 123456789
// Check Digit: 3
```

// Full Number with Check Digit: 1234567893

137. ATM PIN VALIDATOR

```
p ublic class Main {
    // Method to check if the PIN is valid
    public static boolean isValidPin(String pin) {
    // Check if the length is either 4 or 6
    if (pin.length() != 4 && pin.length() != 6) {
    return false;
    }
    // Check if all characters are digits
    for (char c : pin.toCharArray()) {
    if (!Character.isDigit(c)) {
    return false; // Found a non-digit character
    }
    }
    return true; // All checks passed
    }
    public static void main(String[] args) {
    // Example usage
    String[] pins = {"1234", "12345", "a234", "", "123456"};
    for (String pin : pins) {
    System.out.println("PIN '" + pin + "' is valid: " + isValidPin(pin));
    }
```

```
}
}
// PIN '1234' is valid: true
// PIN '12345' is valid: false
// PIN 'a234' is valid: false
// PIN '' is valid: false
// PIN '123456' is valid: true
```

138. CHECK IF A YEAR IS A MAGIC YEAR

```
p ublic class Main {
    // Method to check if a year is a magic year
    public static boolean isMagicYear(int year) {
    String yearStr = String.valueOf(year);
    // Check if the year has exactly 6 digits
    if (yearStr.length() != 6) {
    return false;
    }
    // Extract the month, day, and result from the year string
    int month = Integer.parseInt(yearStr.substring(0, 2));
    int day = Integer.parseInt(yearStr.substring(2, 4));
    int result = Integer.parseInt(yearStr.substring(4, 6));
    // Check if the month multiplied by the day equals the result
    return month * day == result;
    }
    public static void main(String[] args) {
    // Example usage
    int yearToCheck = 1978; // Replace with the year you want to check
    if (isMagicYear(yearToCheck)) {
    System.out.println(yearToCheck + " is a Magic Year! 🎉");
```

```
} else {
System.out.println(yearToCheck + " is not a Magic Year. 😢");
}
}
}
// 1978 is not a Magic Year. 😢
```

139. ENHARMONIC EQUIVALENTS

```
i mport java.util.HashMap ;
    import java.util.Map;
    public class Main {
    // Method to get the enharmonic equivalent of a given note
    public static String getEquivalent(String note) {
    // Define the mapping of notes to their enharmonic equivalents
    Map<String, String> enharmonicMap = new HashMap<>();
    enharmonicMap.put("C#", "Db");
    enharmonicMap.put("Db", "C#");
    enharmonicMap.put("D#", "Eb");
    enharmonicMap.put("Eb", "D#");
    enharmonicMap.put("F#", "Gb");
    enharmonicMap.put("Gb", "F#");
    enharmonicMap.put("G#", "Ab");
    enharmonicMap.put("Ab", "G#");
    enharmonicMap.put("A#", "Bb");
    enharmonicMap.put("Bb", "A#");
    // Notes without enharmonic equivalents map to themselves
    enharmonicMap.put("C", "C");
    enharmonicMap.put("D", "D");
    enharmonicMap.put("E", "E");
```

```
    enharmonicMap.put("F", "F");
    enharmonicMap.put("G", "G");
    enharmonicMap.put("A", "A");
    enharmonicMap.put("B", "B");
    // Return the enharmonic equivalent from the map, default to "Invalid
note"
    return enharmonicMap.getOrDefault(note, "Invalid note");
    }
    public static void main(String[] args) {
    // Example usage
    System.out.println(getEquivalent("D#"));  // Output: Eb
    System.out.println(getEquivalent("Gb"));  // Output: F#
    System.out.println(getEquivalent("Bb"));  // Output: A#
    System.out.println(getEquivalent("C"));   // Output: C
    System.out.println(getEquivalent("H"));   // Output: Invalid note
    }
    }
    // Eb
    // F#
    // A#
    // C
    // Invalid note
```

140. FIND THE AREA OF A REGULAR POLYGON

```
p ublic class Main {
    // Function to calculate the area of a regular polygon
    public static double calculateRegularPolygonArea(int n, double s) {
    double numerator = 1.0 / 4.0 * n * Math.pow(s, 2);
    double denominator = Math.tan(Math.PI / n);
    double area = numerator / denominator;
    return area;
    }
    public static void main(String[] args) {
    int numberOfSides = 6; // Replace with the number of sides of your polygon
    double sideLength = 5.0; // Replace with the length of each side of your polygon
    double polygonArea = calculateRegularPolygonArea(numberOfSides, sideLength);
    System.out.printf("The area of the regular polygon is: %.2f%n", polygonArea);
    }
    }
    // The area of the regular polygon is: 64.95
```

141. CHECK IF A NUMBER IS AN ABUNDANT NUMBER

```
p ublic class Main {
    // Function to calculate the sum of proper divisors
    public static int getProperDivisorsSum(int number) {
    int divisorSum = 0;
    int limit = number / 2; // Proper divisors are less than the number
    for (int i = 1; i <= limit; i++) {
    if (number % i == 0) {
    divisorSum += i; // Add the divisor to the sum
    }
    }
    return divisorSum;
    }
    // Function to check if a number is an abundant number
    public static boolean isAbundantNumber(int number) {
    int divisorsSum = getProperDivisorsSum(number);
    return divisorsSum > number; // Check if the sum of divisors is greater
than the number
    }
    public static void main(String[] args) {
    int numberToCheck = 12; // Replace with the number you want to check
```

```
    if (isAbundantNumber(numberToCheck)) {
    System.out.printf("%d is an Abundant Number! 🎉%n",
numberToCheck);
    } else {
    System.out.printf("%d is not an Abundant Number. 😢%n",
numberToCheck);
    }
    }
    }
    // 12 is an Abundant Number! 🎉
```

142. WASH YOUR HANDS

```
p ublic class Main {
    // Function to calculate the total time spent washing hands
    public static String washHands(int n, int nm) {
    // Total seconds spent washing hands
    int totalSeconds = n * nm * 21;
    // Convert seconds to minutes and seconds
    int minutes = totalSeconds / 60;
    int seconds = totalSeconds % 60;
    return String.format("%d minutes and %d seconds", minutes, seconds);
    }
    public static void main(String[] args) {
    // Example usage
    System.out.println(washHands(8, 7));  // Output: "588 minutes and 0 seconds"
    System.out.println(washHands(0, 0));  // Output: "0 minutes and 0 seconds"
    System.out.println(washHands(7, 9));  // Output: "661 minutes and 30 seconds"
    }
    }
    // 588 minutes and 0 seconds
    // 0 minutes and 0 seconds
```

// 661 minutes and 30 seconds

143. CALCULATE THE EULER'S TOTIENT FUNCTION

```
p ublic class Main {
    // Function to calculate Euler's Totient Function
    public static int eulerTotientFunction(int n) {
    if (n <= 0) {
    throw new IllegalArgumentException("Input must be a positive integer.");
    }
    int result = n; // Initialize result with n
    int p = 2;
    // Iterate through all prime factors of n
    while (p * p <= n) {
    // Check if p is a divisor of n
    if (n % p == 0) {
    // If p divides n, divide n by p as many times as possible
    while (n % p == 0) {
    n /= p;
    }
    result -= result / p; // Apply the formula for prime factors
    }
    p++;
```

```
    }
    // If n is a prime number greater than 1
    if (n > 1) {
    result -= result / n;
    }
    return result; // Return the result
    }
    public static void main(String[] args) {
    // Example usage
    int n = 12;  // Replace this with the number you want to calculate the totient function for
    int result = eulerTotientFunction(n);
    System.out.printf("Euler's Totient Function for %d is: %d%n", n, result);
    }
    }
    // Euler's Totient Function for 12 is: 4
```

144. WHO'S THE OLDEST?

```
i mport java.util.HashMap ;
    import java.util.Map;
    public class Main {
    // Function to find the name of the oldest person
    public static String oldest(Map<String, Integer> people) {
    String oldestName = "";
    int maxAge = 0;
    // Iterate through the map to find the oldest person
    for (Map.Entry<String, Integer> entry : people.entrySet()) {
    String name = entry.getKey();
    int age = entry.getValue();
    if (age > maxAge) {
    maxAge = age;
    oldestName = name;
    }
    }
    return oldestName;
    }
    public static void main(String[] args) {
    // Example usage with first set of people
    Map<String, Integer> people1 = new HashMap<>();
```

```
people1.put("Emma", 71);
people1.put("Jack", 45);
people1.put("Amy", 15);
people1.put("Ben", 29);
System.out.println("The oldest person is: " + oldest(people1));
// Example usage with second set of people
Map<String, Integer> people2 = new HashMap<>();
people2.put("Max", 9);
people2.put("Josh", 13);
people2.put("Sam", 48);
people2.put("Anne", 33);
System.out.println("The oldest person is: " + oldest(people2));
}
}
// The oldest person is: Emma
// The oldest person is: Sam
```

145. DIGITAL CIPHER

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class Main {
    // Function to encode a message using the digital cipher
    public static List<Integer> digitalCipher(String message, int key) {
    // Convert the key to a string and extract its digits
    String keyString = String.valueOf(key);
    List<Integer> keyDigits = new ArrayList<>();
    for (char d : keyString.toCharArray()) {
    keyDigits.add(Character.getNumericValue(d));
    }
    // Prepare the list to hold the encoded values
    List<Integer> encodedValues = new ArrayList<>();
    // Encode the message
    for (int i = 0; i < message.length(); i++) {
    char c = message.charAt(i);
    int letterValue = (c - 'a' + 1); // Calculate the letter value
    int encodedValue = letterValue + keyDigits.get(i % keyDigits.size()); //
Add key digit
    encodedValues.add(encodedValue);
    }
```

```
return encodedValues;
}
public static void main(String[] args) {
// Example usage
List<Integer> encodedMessage1 = digitalCipher("scout", 1939);
List<Integer> encodedMessage2 = digitalCipher("hernando", 1990);
List<Integer> encodedMessage3 = digitalCipher("abella", 100);
System.out.println(encodedMessage1); // [20, 12, 18, 30, 21]
System.out.println(encodedMessage2); // [9, 14, 27, 14, 2, 23, 13, 15]
System.out.println(encodedMessage3); // [2, 2, 5, 13, 12, 1]
}
}
```

146. CHOCOLATE DILEMMA

```
i mport java.util.List ;
    public class ChocolateDilemma {
    // Function to calculate the total area of chocolate pieces
    public static int totalArea(List<int[]> pieces) {
    int totalArea = 0;
    for (int[] piece : pieces) {
    totalArea += piece[0] * piece[1]; // l * w
    }
    return totalArea;
    }
    // Function to check if the chocolate pieces are fair
    public static boolean testFairness(List<int[]> sister1, List<int[]> sister2)
{
    return totalArea(sister1) == totalArea(sister2);
    }
    public static void main(String[] args) {
    // Example usage
    List<int[]> agatha1 = List.of(new int[]{4, 3}, new int[]{2, 4}, new int[]
{1, 2});
    List<int[]> bertha1 = List.of(new int[]{6, 2}, new int[]{4, 2}, new int[]
{1, 1}, new int[]{1, 1});
    System.out.println(testFairness(agatha1, bertha1)); // true
```

```
    List<int[]> agatha2 = List.of(new int[]{1, 2}, new int[]{2, 1});
    List<int[]> bertha2 = List.of(new int[]{2, 2});
    System.out.println(testFairness(agatha2, bertha2)); // true
    List<int[]> agatha3 = List.of(new int[]{1, 2}, new int[]{2, 1});
    List<int[]> bertha3 = List.of(new int[]{2, 2}, new int[]{4, 4});
    System.out.println(testFairness(agatha3, bertha3)); // false
    List<int[]> agatha4 = List.of(new int[]{2, 2}, new int[]{2, 2}, new int[]
{2, 2}, new int[]{2, 2});
    List<int[]> bertha4 = List.of(new int[]{4, 4});
    System.out.println(testFairness(agatha4, bertha4)); // true
    List<int[]> agatha5 = List.of(new int[]{1, 5}, new int[]{6, 3}, new int[]
{1, 1});
    List<int[]> bertha5 = List.of(new int[]{7, 1}, new int[]{2, 2}, new int[]
{1, 1});
    System.out.println(testFairness(agatha5, bertha5)); // false
    }
    }
    // true
    // true
    // false
    // true
    // false
```

147. CALCULATE THE AREA OF A HEXAGON

```
p ublic class HexagonArea {
    // Function to calculate the area of a hexagon
    public static double calculateHexagonArea(double sideLength) {
    // Area of a hexagon formula: (3 * sqrt(3) / 2) * side_length^2
    return (3.0 * Math.sqrt(3) / 2.0) * Math.pow(sideLength, 2);
    }
    public static void main(String[] args) {
    // Example usage
    double sideLength = 5.0; // Replace with the length of a side of your hexagon
    double hexagonArea = calculateHexagonArea(sideLength);
    System.out.printf("The area of the hexagon is: %.2f%n", hexagonArea);
    }
    }
    // The area of the hexagon is: 64.95
```

148. CHECK IF A NUMBER IS A PRONIC NUMBER

```
p ublic class PronicNumberChecker {
    // Function to check if a number is a pronic number
    public static boolean isPronicNumber(int number) {
    for (int i = 0; i <= number; i++) {
    if (i * (i + 1) == number) {
    return true;
    }
    }
    return false;
    }
    public static void main(String[] args) {
    // Example usage
    int pronicNumber = 6; // Replace with the number you want to check
    if (isPronicNumber(pronicNumber)) {
    System.out.println(pronicNumber + " is a Pronic Number!");
    } else {
    System.out.println(pronicNumber + " is not a Pronic Number.");
    }
    int pronicNumber2 = 7; // Another example
    if (isPronicNumber(pronicNumber2)) {
```

```
System.out.println(pronicNumber2 + " is a Pronic Number!");
} else {
System.out.println(pronicNumber2 + " is not a Pronic Number.");
}
}
}
// 6 is a Pronic Number!
// 7 is not a Pronic Number.
```

149. VIRTUAL DAC

```
p ublic class VirtualDAC {
    // Function to convert a digital value to the corresponding voltage level
    public static double vDAC(int digitalValue) {
    // Convert the digital value to the corresponding voltage level
    return (digitalValue / 1023.0) * 5.0;
    }
    public static void main(String[] args) {
    // Example usage
    System.out.printf("%.2f%n", vDAC(0));    // 0.00
    System.out.printf("%.2f%n", vDAC(1023)); // 5.00
    System.out.printf("%.2f%n", vDAC(400));  // 1.96
    }
    }
    // 0.00
    // 5.00
    // 1.96
```

150. FIND THE AREA OF A PENTAGON

```
p ublic class PentagonArea {
    // Function to calculate the area of a pentagon given the side length
    public static double calculatePentagonArea(double sideLength) {
    double sqrt5 = Math.sqrt(5.0);
    double area = (1.0 / 4.0) * Math.sqrt(5.0 * (5.0 + 2.0 * sqrt5)) *
Math.pow(sideLength, 2);
    return area;
    }
    public static void main(String[] args) {
    double sideLength = 4.0; // Example side length
    double pentagonArea = calculatePentagonArea(sideLength);
    System.out.printf("The area of the pentagon is: %.2f%n",
pentagonArea); // 27.53
    }
    }
    // The area of the pentagon is: 27.53
```

151. CHECK IF A NUMBER IS A CUBE NUMBER

```
p ublic class CubeNumberChecker {
    // Function to check if a number is a cube number
    public static boolean isCubeNumber(long number) {
    long cubeRoot = Math.round(Math.cbrt(number)); // Calculate the cube root and round it
    return (cubeRoot * cubeRoot * cubeRoot) == number; // Check if cube of the cube root equals the original number
    }
    public static void main(String[] args) {
    long numberToCheck = 27; // Example number to check
    if (isCubeNumber(numberToCheck)) {
    System.out.printf("%d is a Cube Number! 🎲%n", numberToCheck);
    } else {
    System.out.printf("%d is not a Cube Number. 😢%n", numberToCheck);
    }
    }
    }
    // 27 is a Cube Number! 🎲
```

152. WEEKLY SALARY CALCULATION

```
p ublic class WeeklySalaryCalculator {
    public static int weeklySalary(int[] hours) {
    int totalSalary = 0;
    for (int i = 0; i < hours.length; i++) {
    int hour = hours[i];
    if (i < 5) { // Monday to Friday
    if (hour > 8) {
    totalSalary += 8 * 10; // 8 hours at $10 per hour
    totalSalary += (hour - 8) * 15; // Overtime hours at $15 per hour
    } else {
    totalSalary += hour * 10; // All hours at $10 per hour
    }
    } else { // Saturday and Sunday
    if (hour > 8) {
    totalSalary += 8 * 20; // 8 hours at $20 per hour
    totalSalary += (hour - 8) * 30; // Overtime hours at $30 per hour
    } else {
    totalSalary += hour * 20; // All hours at $20 per hour
    }
    }
    }
```

```
return totalSalary;
}
public static void main(String[] args) {
int[] hours = {8, 8, 8, 8, 8, 0, 0}; // Example input
int salary = weeklySalary(hours);
System.out.printf("Total weekly salary: $%d%n", salary);
}
}
// Total weekly salary: $400
```

153. FIND THE AREA OF A CUBE

```
p ublic class CubeSurfaceAreaCalculator {
    public static double calculateCubeSurfaceArea(double sideLength) {
    return 6.0 * Math.pow(sideLength, 2);
    }
    public static void main(String[] args) {
    double sideLength = 4.0; // Example side length
    double cubeSurfaceArea = calculateCubeSurfaceArea(sideLength);
    System.out.printf("The surface area of the cube is: %.2f%n",
cubeSurfaceArea);
    }
    }
    // The surface area of the cube is: 96.00
```

154. FIND THE AREA OF A CONE

```
p ublic class ConeSurfaceAreaCalculator {
    private static final double PI = Math.PI;
    public static double calculateConeSurfaceArea(double radius, double slantHeight) {
    return PI * radius * (radius + slantHeight);
    }
    public static void main(String[] args) {
    double radius = 3.0; // Replace with the radius of the cone's base
    double slantHeight = 5.0; // Replace with the slant height of the cone
    double surfaceArea = calculateConeSurfaceArea(radius, slantHeight);
    System.out.printf("The surface area of the cone is: %.2f%n", surfaceArea);
    }
    }
    // The surface area of the cone is: 75.40
```

155. CHECK IF A NUMBER IS A HAPPY NUMBER

```
i mport java.util.HashSet ;
    public class HappyNumberChecker {
    public static boolean isHappyNumber(int number) {
    HashSet<Integer> seenNumbers = new HashSet<>();
    int num = number;
    while (num != 1 && !seenNumbers.contains(num)) {
    seenNumbers.add(num);
    num = sumOfSquaredDigits(num);
    }
    return num == 1;
    }
    public static int sumOfSquaredDigits(int number) {
    int sum = 0;
    while (number > 0) {
    int digit = number % 10; // Get the last digit
    sum += digit * digit;     // Add the square of the digit to sum
    number /= 10;             // Remove the last digit
    }
    return sum;
    }
```

```
public static void main(String[] args) {
int number = 19;  // Replace with the number you want to check
if (isHappyNumber(number)) {
System.out.printf("%d is a Happy Number! 😃%n", number);
} else {
System.out.printf("%d is not a Happy Number. 😢%n", number);
}
}
}
// 19 is a Happy Number! 😃
```

156. CALCULATE THE AREA OF A TRIANGULAR PRISM

```
p ublic class TriangularPrismSurfaceArea {
    // Function to calculate the area of a triangular prism
    public static double calculateTriangularPrismSurfaceArea(double a, double b, double c, double height) {
    // Calculate the semi-perimeter
    double s = (a + b + c) / 2.0;
    // Calculate the area of the triangular base using Heron's formula
    double baseArea = Math.sqrt(s * (s - a) * (s - b) * (s - c));
    // Calculate the perimeter of the triangular base
    double perimeter = a + b + c;
    // Calculate the surface area of the triangular prism
    double surfaceArea = baseArea + perimeter * height;
    return surfaceArea;
    }
    public static void main(String[] args) {
    double sideA = 3.0; // Length of side A of the triangular base
    double sideB = 4.0; // Length of side B of the triangular base
    double sideC = 5.0; // Length of side C of the triangular base
    double prismHeight = 6.0; // Height of the triangular prism
    double surfaceArea = calculateTriangularPrismSurfaceArea(sideA, sideB, sideC, prismHeight);
```

```
        System.out.printf("The surface area of the triangular prism is: %.2f%n",
surfaceArea);
    }
    }
    // The surface area of the triangular prism is: 78.00
```

157. FIND ASCII CHARCODE OF INVERSE CASE CHARACTER

```
p ublic class InverseCaseCharCode {
    // Function to find the ASCII charcode of the inverse case character
    public static int counterpartCharCode(char c) {
    if (Character.isUpperCase(c)) {
    // Convert uppercase to lowercase
    char lower = Character.toLowerCase(c);
    return (int) lower; // Return ASCII code of the lowercase character
    } else if (Character.isLowerCase(c)) {
    // Convert lowercase to uppercase
    char upper = Character.toUpperCase(c);
    return (int) upper; // Return ASCII code of the uppercase character
    } else {
    // Return the ASCII code of the character if it's not an alphabet
    return (int) c;
    }
    }
    public static void main(String[] args) {
    char[] testCases = {'A', 'a', '1', '#'};
    for (char test : testCases) {
```

```
    System.out.printf("Char: %c, ASCII Code: %d%n", test,
counterpartCharCode(test));
    }
    }
    }
    // Char: A, ASCII Code: 97
    // Char: a, ASCII Code: 65
    // Char: 1, ASCII Code: 49
    // Char: #, ASCII Code: 35
```

158. SOLVE A LINEAR EQUATION

```
p ublic class LinearEquationSolver {
    public static int solveEquation(String equation) {
    String[] parts = equation.split("=");
    String leftSide = parts[0].trim();
    int rightSide = Integer.parseInt(parts[1].trim());
    char operation;
    int num;
    if (leftSide.contains("+")) {
    String[] leftParts = leftSide.split("\\+");
    operation = '+';
    num = Integer.parseInt(leftParts[1].trim());
    } else {
    String[] leftParts = leftSide.split("-");
    operation = '-';
    num = Integer.parseInt(leftParts[1].trim());
    }
    switch (operation) {
    case '+':
    return rightSide - num;
    case '-':
    return rightSide + num;
```

```
default:
throw new IllegalArgumentException("Unsupported operation");
}
}
public static void main(String[] args) {
System.out.println(solveEquation("x + 43 = 50"));  // Outputs: 7
System.out.println(solveEquation("x - 9 = 10"));   // Outputs: 19
System.out.println(solveEquation("x + 300 = 100")); // Outputs: -200
}
}
// 7
// 19
// -200
```

159. FACTORIZE A NUMBER

```
i mport java.util.ArrayList ;
    import java.util.Collections;
    import java.util.List;
    public class FactorizeNumber {
    public static List<Integer> factorize(int n) {
    List<Integer> factors = new ArrayList<>();
    int sqrtN = (int) Math.sqrt(n);
    for (int i = 1; i <= sqrtN; i++) {
    if (n % i == 0) {
    factors.add(i);
    if (i != n / i) {
    factors.add(n / i);
    }
    }
    }
    Collections.sort(factors);
    return factors;
    }
    public static void main(String[] args) {
    int num = 12;
    System.out.println(factorize(num));  // Outputs: [1, 2, 3, 4, 6, 12]
```

```
}
}
// [1, 2, 3, 4, 6, 12]
```

160. CHECK IF A NUMBER IS AN AUTOMORPHIC NUMBER

```
p ublic class AutomorphicNumber {
    public static boolean isAutomorphicNumber(long number) {
    long square = number * number;
    String numberStr = Long.toString(number);
    String squareStr = Long.toString(square);
    return squareStr.endsWith(numberStr);
    }
    public static void main(String[] args) {
    long automorphicNumber = 25; // Replace with the number you want to check
    if (isAutomorphicNumber(automorphicNumber)) {
    System.out.printf("%d is an Automorphic Number!%n", automorphicNumber);
    } else {
    System.out.printf("%d is not an Automorphic Number.%n", automorphicNumber);
    }
    }
    }
    // 25 is an Automorphic Number!
```

161. CALCULATE THE AREA OF A PYRAMID

```
p ublic class PyramidSurfaceArea {
    // Function to calculate the surface area of a pyramid
    public static double calculatePyramidSurfaceArea(double sideLength, double slantHeight) {
    double baseArea = Math.pow(sideLength, 2);
    double lateralArea = 4.0 * sideLength * slantHeight / 2.0; // Each triangular face has area (base * slantHeight) / 2
    return baseArea + lateralArea;
    }
    public static void main(String[] args) {
    double sideLength = 4.0;  // Length of a side of the pyramid's base
    double slantHeight = 5.0;  // Slant height of the pyramid
    double pyramidSurfaceArea = calculatePyramidSurfaceArea(sideLength, slantHeight);
    System.out.printf("The surface area of the pyramid is: %.2f%n", pyramidSurfaceArea);
    }
    }
    // The surface area of the pyramid is: 56.00
```

162. CHECK IF A NUMBER IS A SMITH–MORRA GAMBIT NUMBER

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class SmithMorraGambitNumber {
    // Function to calculate the digit sum of a number
    public static int digitSum(int n) {
    int sum = 0;
    while (n > 0) {
    sum += n % 10;
    n /= 10;
    }
    return sum;
    }
    // Function to perform prime factorization of a number
    public static List<Integer> primeFactorization(int n) {
    List<Integer> factors = new ArrayList<>();
    for (int i = 2; i * i <= n; i++) {
    while (n % i == 0) {
    factors.add(i);
    n /= i;
    }
```

```
    }
    if (n > 1) {
    factors.add(n);
    }
    return factors;
    }
    // Function to check if a number is a Smith–Morra Gambit Number
    public static boolean isSmithMorraGambitNumber(int number) {
    List<Integer> factors = primeFactorization(number);
    int sumOfDigitsOfFactors = 0;
    for (int factor : factors) {
    sumOfDigitsOfFactors += digitSum(factor);
    }
    return sumOfDigitsOfFactors == digitSum(number);
    }
    public static void main(String[] args) {
    int number = 22; // Example number
    System.out.printf("Is %d a Smith–Morra Gambit Number? %b%n",
number, isSmithMorraGambitNumber(number));
    }
    }
    // Is 22 a Smith–Morra Gambit Number? true
```

163. CHECK IF A NUMBER IS A SOLITARY NUMBER

```
p ublic class SolitaryNumber {
    // Function to calculate the sum of proper divisors of a number
    public static int getProperDivisorsSum(int number) {
    int sum = 1; // Start with 1 as every number is divisible by 1
    for (int i = 2; i <= Math.sqrt(number); i++) {
    if (number % i == 0) {
    sum += i;
    if (i != number / i) {
    sum += number / i;
    }
    }
    }
    return sum;
    }
    // Function to check if a number is a Solitary Number
    public static boolean isSolitaryNumber(int number) {
    int sumOfDivisors = getProperDivisorsSum(number);
    return number != sumOfDivisors;
    }
    public static void main(String[] args) {
```

```
int solitaryNumber = 28; // Example number, replace as needed
if (isSolitaryNumber(solitaryNumber)) {
System.out.printf("%d is a Solitary Number!%n", solitaryNumber);
} else {
System.out.printf("%d is not a Solitary Number.%n", solitaryNumber);
}
}
}
// 28 is not a Solitary Number!
```

164. BASIC TOWER OF HANOI PUZZLE

```
p ublic class TowerOfHanoi {
    // Recursive function to solve the Tower of Hanoi puzzle
    public static void towerOfHanoi(int n, char source, char auxiliary, char
target) {
    if (n == 1) {
    System.out.printf("Move disk 1 from %c to %c%n", source, target);
    return;
    }
    towerOfHanoi(n - 1, source, target, auxiliary);
    System.out.printf("Move disk %d from %c to %c%n", n, source, target);
    towerOfHanoi(n - 1, auxiliary, source, target);
    }
    public static void main(String[] args) {
    int numberOfDisks = 3;
    char sourcePeg = 'A';
    char auxiliaryPeg = 'B';
    char targetPeg = 'C';
    System.out.printf("Tower of Hanoi solution for %d disks:%n",
numberOfDisks);
    towerOfHanoi(numberOfDisks, sourcePeg, auxiliaryPeg, targetPeg);
    }
```

```
}
// Tower of Hanoi solution for 3 disks:
// Move disk 1 from A to C
// Move disk 2 from A to B
// Move disk 1 from C to B
// Move disk 3 from A to C
// Move disk 1 from B to A
// Move disk 2 from B to C
// Move disk 1 from A to C
```

165. CALCULATE THE AREA OF A FRUSTUM

```
p ublic class FrustumSurfaceArea {
    // Method to calculate the surface area of a frustum
    public static double surfaceAreaOfFrustum(double r1, double r2, double
l) {
    double surfaceArea = Math.PI * (r1 + r2) * l + Math.PI * Math.pow(r1,
2) + Math.PI * Math.pow(r2, 2);
    return surfaceArea;
    }
    public static void main(String[] args) {
    double r1 = 4.0; // Top radius
    double r2 = 8.0; // Bottom radius
    double l = 6.0;  // Slant height
    double area = surfaceAreaOfFrustum(r1, r2, l);
    System.out.printf("Surface Area of the frustum is: %.2f%n", area);
    }
    }
    // Surface Area of the frustum is: 477.52
```

166. CHECK IF A NUMBER IS A MOTZKIN NUMBER

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class MotzkinNumberChecker {
    // Method to check if a number is a Motzkin number
    public static boolean isMotzkinNumber(int number) {
    if (number == 0) {
    return true;
    }
    List<Double> motzkinNumbers = new ArrayList<>();
    motzkinNumbers.add(1.0);
    motzkinNumbers.add(1.0);
    for (int n = 2; n <= number; n++) {
    double nextMotzkin = ((2.0 * n + 1.0) * motzkinNumbers.get(n - 1) +
    (3.0 * n - 3.0) * motzkinNumbers.get(n - 2)) /
    (n + 2.0);
    motzkinNumbers.add(nextMotzkin);
    }
    return motzkinNumbers.contains((double) number);
    }
    public static void main(String[] args) {
```

```
int[] testNumbers = {1, 2, 5, 6, 11};
for (int number : testNumbers) {
if (isMotzkinNumber(number)) {
System.out.printf("%d is a Motzkin Number!%n", number);
} else {
System.out.printf("%d is not a Motzkin Number.%n", number);
}
}
}
}
// 1 is a Motzkin Number!
// 2 is a Motzkin Number!
// 5 is not a Motzkin Number.
// 6 is not a Motzkin Number.
// 11 is not a Motzkin Number.
```

167. SWAPPING TWO BY TWO

```
p ublic class SwapTwoByTwo {
    // Method to swap characters in pairs
    public static String swapTwo(String s) {
    StringBuilder result = new StringBuilder();
    char[] chars = s.toCharArray();
    int i = 0;
    while (i + 3 < chars.length) {
    result.append(chars[i + 2]); // Swap 3rd character
    result.append(chars[i + 3]); // Swap 4th character
    result.append(chars[i]);      // Original 1st character
    result.append(chars[i + 1]);  // Original 2nd character
    i += 4; // Move to the next set of characters
    }
    // Append any remaining characters
    result.append(s.substring(i));
    return result.toString();
    }
    public static void main(String[] args) {
    String[] examples = {
    "ABCDEFGH",
    "AABBCCDDEEFF",
```

```
    "munchkins",
    "FFGGHHI"
    };
    for (String example : examples) {
    System.out.printf("Original: %s, Swapped: %s%n", example,
swapTwo(example));
    }
    }
    }
    // Original: ABCDEFGH, Swapped: CDABGHEF
    // Original: AABBCCDDEEFF, Swapped: BBAADDCCFFEE
    // Original: munchkins, Swapped: ncmuinhks
    // Original: FFGGHHI, Swapped: GGFFHHI
```

168. EXTRACT A WORD FROM A SENTENCE

```
p ublic class RemoveWordFromSentence {
    // Method to remove a word from a sentence
    public static String removeWord(String sentence, String word) {
    // Split the sentence into words
    String[] words = sentence.split(" ");
    StringBuilder result = new StringBuilder();
    // Iterate over the words and append those that are not equal to the word to remove
    for (String w : words) {
    if (!w.equals(word)) {
    result.append(w).append(" ");
    }
    }
    // Convert StringBuilder to String and trim any trailing spaces
    return result.toString().trim();
    }
    public static void main(String[] args) {
    String sentence1 = "One two three four";
    String word1 = "two";
    String result1 = removeWord(sentence1, word1);
```

```
System.out.println(result1); // Output: "One three four"
String sentence2 = "Bob has a kid";
String word2 = "kid";
String result2 = removeWord(sentence2, word2);
System.out.println(result2); // Output: "Bob has a"
}
}
// One three four
// Bob has a
```

169. BASIC CHATBOT

```
i mport java.util.Scanner ;
    public class BasicChatbot {
    // Method to respond based on the user's message
    public static String chatbot(String message) {
    message = message.toLowerCase(); // Convert message to lowercase
    if (message.contains("hello")) {
    return "Hello! How can I help you?";
    } else if (message.contains("how are you")) {
    return "I am just a computer program, but thanks for asking!";
    } else if (message.contains("bye")) {
    return "Goodbye!";
    } else {
    return "I didn't understand that. Can you please rephrase?";
    }
    }
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Chatbot: Hello! How can I help you today?");
    while (true) {
    System.out.print("You: ");
    String userInput = scanner.nextLine().trim();
```

```
if (userInput.isEmpty()) {
continue; // Skip empty input
}
if (userInput.equalsIgnoreCase("bye")) {
System.out.println("Chatbot: Goodbye!");
break; // Exit the loop if the user says "bye"
}
System.out.println("Chatbot: " + chatbot(userInput));
}
scanner.close(); // Close the scanner
}
}
// Chatbot: Hello! How can I help you today?
// You: Hello
// Chatbot: Hello! How can I help you?
// You: How are you?
// Chatbot: I am just a computer program, but thanks for asking!
// You: bye
// Chatbot: Goodbye!
```

170. BACKSPACE ATTACK

```
p ublic class BackspaceAttack {
    // Method to simulate the backspace action
    public static String erase(String input) {
    StringBuilder result = new StringBuilder();
    for (char c : input.toCharArray()) {
    if (c == '#') {
    if (result.length() > 0) {
    result.deleteCharAt(result.length() - 1); // Remove the last character if it
exists
    }
    } else {
    result.append(c); // Append the current character
    }
    }
    return result.toString();
    }
    public static void main(String[] args) {
    String[] examples = {
    "he##l#hel#llo",
    "major# spar##ks",
    "si###t boy",
```

```
    "####",
    };
    for (String example : examples) {
    System.out.printf("Input: '%s', Output: '%s'%n", example,
erase(example));
    }
    }
    }
    // Input: 'he##l#hel#llo', Output: 'hello'
    // Input: 'major# spar##ks', Output: 'majo spks'
    // Input: 'si###t boy', Output: 't boy'
    // Input: '####', Output: ''
```

171. COUNTER

```
p ublic class Counter {
    private int count;
    // Constructor to initialize the counter
    public Counter() {
    this.count = 0;
    }
    // Method to increment the counter
    public void increment() {
    count++;
    System.out.println("Counter: " + count);
    }
    // Method to decrement the counter
    public void decrement() {
    count—;
    System.out.println("Counter: " + count);
    }
    // Method to reset the counter
    public void reset() {
    count = 0;
    System.out.println("Counter reset to 0");
    }
```

```
// Main method to test the Counter class
public static void main(String[] args) {
Counter counter = new Counter();
counter.increment(); // Counter: 1
counter.increment(); // Counter: 2
counter.decrement(); // Counter: 1
counter.reset();     // Counter reset to 0
}
}
// Counter: 1
// Counter: 2
// Counter: 1
// Counter reset to 0
```

172. PHONE NUMBER WORD DECODER

```
i mport java.util.HashMap ;
    public class PhoneNumberDecoder {
    public static String textToNum(String phone) {
    HashMap<Character, Character> map = new HashMap<>();
    // Mapping letters to corresponding digits
    map.put('A', '2'); map.put('B', '2'); map.put('C', '2');
    map.put('D', '3'); map.put('E', '3'); map.put('F', '3');
    map.put('G', '4'); map.put('H', '4'); map.put('I', '4');
    map.put('J', '5'); map.put('K', '5'); map.put('L', '5');
    map.put('M', '6'); map.put('N', '6'); map.put('O', '6');
    map.put('P', '7'); map.put('Q', '7'); map.put('R', '7'); map.put('S', '7');
    map.put('T', '8'); map.put('U', '8'); map.put('V', '8');
    map.put('W', '9'); map.put('X', '9'); map.put('Y', '9'); map.put('Z', '9');
    StringBuilder result = new StringBuilder();
    for (char c : phone.toCharArray()) {
    // Check if character is a letter
    if (map.containsKey(Character.toUpperCase(c))) {
    // Append the corresponding number
    result.append(map.get(Character.toUpperCase(c)));
    } else {
```

```
// Otherwise, append the character itself
result.append(c);
}
}
return result.toString();
}
public static void main(String[] args) {
String[] examples = {
"123-647-EYES",
"(325)444-TEST",
"653-TRY-THIS",
"435-224-7613"
};
for (String example : examples) {
System.out.println(textToNum(example));
}
}
}
// 123-647-3937
// (325)444-8378
// 653-879-8447
// 435-224-7613
```

173. COIN CO-OPERATION

```
p ublic class CoinCoOperation {
    public static int[] getCoinBalances(String[] p1Choices, String[]
p2Choices) {
    int coinsP1 = 3;
    int coinsP2 = 3;
    for (int i = 0; i < p1Choices.length; i++) {
    String c1 = p1Choices[i];
    String c2 = p2Choices[i];
    if (c1.equals("share") && c2.equals("share")) {
    coinsP1 += 2;
    coinsP2 += 2;
    } else if (c1.equals("share") && c2.equals("steal")) {
    coinsP1 -= 1;
    coinsP2 += 3;
    } else if (c1.equals("steal") && c2.equals("share")) {
    coinsP1 += 3;
    coinsP2 -= 1;
    }
    }
    return new int[]{coinsP1, coinsP2};
    }
```

```
public static void main(String[] args) {
String[] p1Choices = {"share", "share", "share"};
String[] p2Choices = {"steal", "share", "steal"};
int[] balances = getCoinBalances(p1Choices, p2Choices);
System.out.println("Person 1 balance: " + balances[0]);
System.out.println("Person 2 balance: " + balances[1]);
}
}
// Person 1 balance: 3
// Person 2 balance: 11
```

174. VALIDATE PIN

```
i mport java.util.regex .Pattern;
    public class ValidatePIN {
    public static boolean validate(String pin) {
    // Regular expression to match the valid PIN patterns
    String regex = "^\\d{4}(\\d{2})?$";
    return Pattern.matches(regex, pin);
    }
    public static void main(String[] args) {
    // Test cases
    System.out.println(validate("121317")); // true
    System.out.println(validate("1234"));   // true
    System.out.println(validate("45135"));  // false
    System.out.println(validate("89abc1")); // false
    System.out.println(validate("900876")); // true
    System.out.println(validate(" 4983"));  // false
    }
    }
    // true
    // true
    // false
    // false
```

```
// true
// false
```

175. RANDOM NUMBER GENERATOR

```
i mport java.util.Random ;
    public class RandomNumberGenerator {
    public static int generateRandomNumber(int minVal, int maxVal) {
    Random rng = new Random();
    return rng.nextInt(maxVal - minVal + 1) + minVal;
    }
    public static void main(String[] args) {
    int randomNumber = generateRandomNumber(1, 10);
    System.out.println("Random Number: " + randomNumber);
    }
    }
    // Random Number: 5
```

176. SEVEN BOOM!

```
p ublic class SevenBoom {
    public static String sevenBoom(int[] numbers) {
    for (int num : numbers) {
    if (String.valueOf(num).contains("7")) {
    return "Boom!";
    }
    }
    return "there is no 7 in the array";
    }
    public static void main(String[] args) {
    // Test examples
    int[][] examples = {
    {1, 2, 3, 4, 5, 6, 7},
    {8, 6, 33, 100},
    {2, 55, 60, 97, 86},
    };
    for (int[] numbers : examples) {
    System.out.println(sevenBoom(numbers));
    }
    }
    }
```

```
// Boom!
// there is no 7 in the array
// Boom!
```

177. CAPITALIZE THE LAST LETTER

```
p ublic class CapitalizeLastLetter {
    public static String capLast(String input) {
    StringBuilder result = new StringBuilder();
    // Split the input string into words
    String[] words = input.split("\\s+");
    for (String word : words) {
    if (word.length() > 0) {
    // Capitalize the last letter
    char lastChar = word.charAt(word.length() - 1);
    String capitalizedWord = word.substring(0, word.length() - 1) +
Character.toUpperCase(lastChar);
    result.append(capitalizedWord).append(" "); // Append the modified
word to result
    }
    }
    // Convert StringBuilder to String and trim the trailing space
    return result.toString().trim();
    }
    public static void main(String[] args) {
    // Test examples
    String[] examples = {
```

```
"hello",
"My Name Is Example",
"HELp THe LASt LETTERs CAPITALISe",
};
for (String example : examples) {
System.out.println(capLast(example));
}
}
}
// hellO
// MY NamE IS ExamplE
// HELp THE LASt LETTERs CAPITALIS
```

178. DICE ROLLING SIMULATOR

```
i mport java.util.Random ;
    public class DiceRollingSimulator {
    // Method to roll the dice
    public static void rollDice() {
    Random random = new Random(); // Create a Random object
    int result = random.nextInt(6) + 1; // Generate a random number
between 1 and 6
    System.out.println("You rolled a " + result);
    }
    public static void main(String[] args) {
    rollDice(); // Simulate rolling the dice
    }
    }
    // You rolled a 3
```

179. SECONDS TO TIME CONVERTER

```
p ublic class SecondsToTimeConverter {
    // Method to convert seconds to hours, minutes, and seconds
    public static void convertSecondsToTime(int seconds) {
    int hours = seconds / 3600; // Calculate hours
    int minutes = (seconds % 3600) / 60; // Calculate minutes
    int remainingSeconds = seconds % 60; // Calculate remaining seconds
    System.out.printf("Time: %d hours, %d minutes, %d seconds%n",
hours, minutes, remainingSeconds);
    }
    public static void main(String[] args) {
    int inputSeconds = 3665; // Example input
    convertSecondsToTime(inputSeconds); // Convert and display the time
    }
    }
    // Time: 1 hours, 1 minutes, 5 seconds
```

180. BAR CHART GENERATOR

```
i mport java.util.Arrays ;
    public class BarChartGenerator {
    // Method to generate a bar chart from data
    public static void generateBarChart(int[] data) {
    if (data.length == 0) {
    System.out.println("No data to display.");
    return;
    }
    // Find the maximum value in the data
    int maxValue = Arrays.stream(data).max().orElse(0);
    // Generate and print the bar chart
    for (int value : data) {
    // Calculate the length of the bar based on the maximum value
    int barLength = (int) Math.round((double) value / maxValue * 20);
    String bar = "".repeat(barLength) + " ".repeat(20 - barLength);
    System.out.printf("%d | %s%n", value, bar);
    }
    }
    public static void main(String[] args) {
    int[] chartData = {5, 8, 12, 4, 6}; // Example data
    generateBarChart(chartData); // Generate and display the bar chart
```

```
}
}
// 5 | [redacted]
// 8 | [redacted]
// 12 | [redacted]
// 4 | [redacted]
// 6 | [redacted]
```

181. RIGHT-ANGLED TRIANGLE PATTERN

```
p ublic class RightAngledTriangle {
    // Method to generate a right-angled triangle pattern
    public static void generateRightAngledTriangle(int height) {
    for (int i = 1; i <= height; i++) {
    String stars = "*".repeat(i); // Generate stars for the current row
    System.out.println(stars); // Print the current row
    }
    }
    public static void main(String[] args) {
    int triangleHeight = 5; // Example height
    generateRightAngledTriangle(triangleHeight); // Generate and print the
triangle
    }
    }
    // *
    // **
    // ***
    // ****
    // *****
```

182. POSITIVE COUNT / NEGATIVE SUM

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class PositiveCountNegativeSum {
    // Method to count positives and sum negatives
    public static List<Integer> countPositivesSumNegatives(int[] numbers)
{
    int countPositives = 0;
    int sumNegatives = 0;
    for (int num : numbers) {
    if (num > 0) {
    countPositives++;
    } else if (num < 0) {
    sumNegatives += num;
    }
    }
    List<Integer> result = new ArrayList<>();
    if (countPositives > 0 || sumNegatives < 0) {
    result.add(countPositives);
    result.add(sumNegatives);
    }
```

```
    return result; // Return the result list
    }
    public static void main(String[] args) {
    int[] example1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -11, -12, -13, -14, -15};
    int[] example2 = {92, 6, 73, -77, 81, -90, 99, 8, -85, 34};
    int[] example3 = {91, -4, 80, -73, -28};
    int[] example4 = {}; // Empty array
    System.out.println(countPositivesSumNegatives(example1)); // [10,
-65]
    System.out.println(countPositivesSumNegatives(example2)); // [7,
-252]
    System.out.println(countPositivesSumNegatives(example3)); // [2,
-105]
    System.out.println(countPositivesSumNegatives(example4)); // []
    }
    }
    // [10, -65]
    // [7, -252]
    // [2, -105]
    // []
```

183. NUMBER PYRAMID GENERATOR

```
p ublic class NumberPyramidGenerator {
    // Method to generate the number pyramid
    public static void generateNumberPyramid(int height) {
    for (int i = 1; i <= height; i++) {
    // Create leading spaces
    String spaces = " ".repeat(height - i);
    // Generate the numbers for the current row
    StringBuilder numbers = new StringBuilder();
    for (int num = 1; num <= i; num++) {
    numbers.append(num).append(" ");
    }
    // Print the current row
    System.out.println(spaces + numbers.toString().trim());
    }
    }
    public static void main(String[] args) {
    int pyramidHeight = 4; // Example height
    generateNumberPyramid(pyramidHeight);
    }
    }
    //    1
```

```
//   1 2
//  1 2 3
// 1 2 3 4
```

184. DIAMOND PATTERN GENERATOR

```
p ublic class DiamondPatternGenerator {
    // Method to generate the diamond pattern
    public static void generateDiamondPattern(int height) {
    // Check if height is an odd positive integer
    if (height % 2 == 0 || height < 1) {
    System.out.println("Height must be an odd positive integer.");
    return;
    }
    int midpoint = (height + 1) / 2;
    for (int i = 1; i <= height; i++) {
    int spaces = Math.abs(midpoint - i); // Calculate spaces
    int stars = height - 2 * spaces; // Calculate stars
    // Print leading spaces and stars
    System.out.printf("%" + (spaces + stars) + "s%n", "*".repeat(stars));
    }
    }
    public static void main(String[] args) {
    generateDiamondPattern(5); // Example height
    }
    }
    //   *
```

```
//  ***
// *****
//  ***
//   *
```

185. CHECK IF THE BRICK FITS THROUGH THE HOLE

```
p ublic class BrickFitChecker {
    // Method to check if brick dimensions fit through the hole dimensions
    public static boolean doesBrickFit(int a, int b, int c, int w, int h) {
    // Check all possible orientations of the brick
    return fits(a, b, w, h) || fits(a, c, w, h) || fits(b, c, w, h);
    }
    // Helper method to check if the brick can fit in the hole
    private static boolean fits(int brickWidth, int brickHeight, int
holeWidth, int holeHeight) {
    return (brickWidth <= holeWidth && brickHeight <= holeHeight) ||
    (brickWidth <= holeHeight && brickHeight <= holeWidth);
    }
    public static void main(String[] args) {
    // Test cases
    System.out.println(doesBrickFit(1, 1, 1, 1, 1)); // true
    System.out.println(doesBrickFit(1, 2, 1, 1, 1)); // true
    System.out.println(doesBrickFit(1, 2, 2, 1, 1)); // false
    }
    }
    // true
```

```
// true
// false
```

186. COUNTDOWN TIMER

```
i mport java.util.Scanner ;
    public class CountdownTimer {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter countdown time in seconds: ");
    // Read user input
    if (scanner.hasNextLong()) {
    long time = scanner.nextLong();
    // Start countdown
    while (time > 0) {
    System.out.println("Time Remaining: " + time + " seconds");
    try {
    Thread.sleep(1000); // Sleep for 1 second
    } catch (InterruptedException e) {
    System.out.println("Timer was interrupted.");
    }
    time—;
    }
    System.out.println("Countdown completed!");
    } else {
    System.out.println("Invalid input.");
```

```
}
scanner.close();
}
}
// Enter countdown time in seconds: 3
// Time Remaining: 3 seconds
// Time Remaining: 2 seconds
// Time Remaining: 1 seconds
// Countdown completed!
```

187. STATE NAMES AND ABBREVIATIONS

```
i mport java.util.ArrayList ;
    import java.util.Arrays;
    import java.util.List;
    public class StateNames {
    public static List<String> filterStateNames(List<String> states, String
category) {
    // Full state names
    String[] fullStateNames = {
    "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado",
"Connecticut",
    "Delaware", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois",
"Indiana", "Iowa",
    "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland",
"Massachusetts", "Michigan",
    "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska",
"Nevada", "New Hampshire",
    "New Jersey", "New Mexico", "New York", "North Carolina", "North
Dakota", "Ohio", "Oklahoma",
    "Oregon", "Pennsylvania", "Rhode Island", "South Carolina", "South
Dakota", "Tennessee",
    "Texas", "Utah", "Vermont", "Virginia", "Washington", "West Virginia",
"Wisconsin", "Wyoming"
```

```
    };
    List<String> result = new ArrayList<>();
    for (String state : states) {
    switch (category) {
    case "abb":
    if (state.length() == 2) {
    result.add(state);
    }
    break;
    case "full":
    if (Arrays.asList(fullStateNames).contains(state)) {
    result.add(state);
    }
    break;
    default:
    break;
    }
    }
    return result;
    }
    public static void main(String[] args) {
    List<String> states1 = Arrays.asList("Arizona", "CA", "NY",
"Nevada");
```

```
    List<String> states2 = Arrays.asList("MT", "NJ", "TX", "ID", "IL");
    System.out.println(filterStateNames(states1, "abb")); // → [CA, NY]
    System.out.println(filterStateNames(states1, "full")); // → [Arizona,
Nevada]
    System.out.println(filterStateNames(states2, "abb")); // → [MT, NJ, TX,
ID, IL]
    System.out.println(filterStateNames(states2, "full")); // → []
    }
    }
    // [CA, NY]
    // [Arizona, Nevada]
    // [MT, NJ, TX, ID, IL]
    // []
```

188. FUNCTIONINATOR 8000

```
p ublic class Functioninator8000 {
    public static String inatorInator(String word) {
    // Helper function to check if a character is a consonant
    boolean isConsonant(char c) {
    return Character.isAlphabetic(c) && "aeiouAEIOU".indexOf(c) == -1;
    }
    char lastChar = word.length() > 0 ? word.charAt(word.length() - 1) : ' ';
    // Determine the suffix based on the last character
    String suffix = isConsonant(lastChar) ? "inator" : "-inator";
    // Calculate the length and format it
    String lengthSuffix = String.format("%d000", word.length());
    // Combine the parts into the final string
    return String.format("%s%s %s", word, suffix, lengthSuffix);
    }
    public static void main(String[] args) {
    String[] testCases = {"Shrink", "Doom", "EvilClone"};
    for (String word : testCases) {
    System.out.println(inatorInator(word));
    }
    }
    }
```

// Shrinkinator 6000

// Doominator 4000

// EvilClone-inator 9000

189. PAGES IN A BOOK

```
p ublic class PagesInABook {
    // Method to check if a number is a perfect square
    private static boolean isPerfectSquare(long x) {
    long s = (long) Math.sqrt(x);
    return s * s == x;
    }
    // Method to determine if the total pages can form a perfect book
    public static boolean pagesInBook(long total) {
    if (total <= 0) {
    return false;
    }
    long discriminant = 1 + 8 * total;
    if (!isPerfectSquare(discriminant)) {
    return false;
    }
    long sqrtDisc = (long) Math.sqrt(discriminant);
    long n = (sqrtDisc - 1) / 2;
    // Verify that n is a positive integer and satisfies the original condition
    return n > 0 && n * (n + 1) / 2 == total;
    }
    public static void main(String[] args) {
    long[] testCases = {5, 4005, 9453};
```

```
for (long total : testCases) {
System.out.printf("Total pages %d: %s%n", total, pagesInBook(total));
}
}
}
// Total pages 5: false
// Total pages 4005: true
// Total pages 9453: true
```

190. HIGHEST DIGIT

```
p ublic class HighestDigit {
    // Method to find the highest digit in a number
    public static int highestDigit(int n) {
    int highest = 0;
    // Convert the number to a string to iterate over each character
    String numberStr = Integer.toString(n);
    for (char c : numberStr.toCharArray()) {
    // Convert the character back to an integer
    int digit = Character.getNumericValue(c);
    // Update the highest digit found
    if (digit > highest) {
    highest = digit;
    }
    }
    return highest;
    }
    public static void main(String[] args) {
    int[] testCases = {4666, 544, 379, 2, 377401};
    for (int caseNum : testCases) {
    System.out.printf("Highest digit in %d: %d%n", caseNum,
highestDigit(caseNum));
```

```
}
}
}
```

```
Highest digit in 4666: 6
Highest digit in 544: 5
Highest digit in 379: 9
Highest digit in 2: 2
Highest digit in 377401: 7
```

191. VIDEO LENGTH IN SECONDS

```
p ublic class VideoLength {
    // Method to convert video length from "MM:SS" format to total seconds
    public static int minutesToSeconds(String videoLength) {
    String[] parts = videoLength.split(":");
    // Check for valid format
    if (parts.length != 2) {
    return -1; // Invalid format
    }
    // Parse minutes
    int minutes;
    try {
    minutes = Integer.parseInt(parts[0]);
    } catch (NumberFormatException e) {
    return -1; // Invalid minutes value
    }
    // Parse seconds
    int seconds;
    try {
    seconds = Integer.parseInt(parts[1]);
    } catch (NumberFormatException e) {
```

```
    return -1; // Invalid seconds value
    }
    // Validate seconds
    if (seconds >= 60) {
    return -1; // Invalid seconds value
    }
    return minutes * 60 + seconds; // Return total seconds
    }
    public static void main(String[] args) {
    String[] testCases = {"01:00", "13:56", "10:60", "121:49", "invalid"};
    for (String caseStr : testCases) {
    System.out.printf("Video Length '%s': %d seconds%n", caseStr,
minutesToSeconds(caseStr));
    }
    }
    }
    // Video Length '01:00': 60 seconds
    // Video Length '13:56': 836 seconds
    // Video Length '10:60': -1 seconds
    // Video Length '121:49': 7309 seconds
    // Video Length 'invalid': -1 seconds
```

192. COUNT LETTERS IN A WORD SEARCH

```
p ublic class LetterCounter {
    // Method to count occurrences of a letter in a 2D grid
    public static int letterCounter(String[][] grid, String letter) {
    int count = 0;
    // Iterate through each row and column in the grid
    for (String[] row : grid) {
    for (String ch : row) {
    if (ch.equals(letter)) {
    count++; // Increment count if letter matches
    }
    }
    }
    return count; // Return the total count
    }
    public static void main(String[] args) {
    String[][] grid = {
    {"D", "E", "Y", "H", "A", "D"},
    {"C", "B", "Z", "Y", "J", "K"},
    {"D", "B", "C", "A", "M", "N"},
    {"F", "G", "G", "R", "S", "R"},
```

```
        {"V", "X", "H", "A", "S", "S"}
        };
        String letter = "D";
        System.out.printf("The letter '%s' appears %d times.%n", letter,
letterCounter(grid, letter));
        letter = "H";
        System.out.printf("The letter '%s' appears %d times.%n", letter,
letterCounter(grid, letter));
        }
        }
        // The letter 'D' appears 3 times.
        // The letter 'H' appears 2 times.
```

193. FIND THE OTHER TWO SIDE LENGTHS

```
p ublic class TriangleSides {
    private static final double SQRT_3 = 1.7320508075688772;
    // Method to calculate the lengths of the other two sides
    public static double[] otherSides(double shortestSide) {
    double hypotenuse = 2.0 * shortestSide; // Calculate hypotenuse
    double otherSide = shortestSide * SQRT_3; // Calculate the other side
    // Rounding to 2 decimal places
    hypotenuse = Math.round(hypotenuse * 100.0) / 100.0;
    otherSide = Math.round(otherSide * 100.0) / 100.0;
    return new double[] { hypotenuse, otherSide };
    }
    public static void main(String[] args) {
    double[] testCases = { 1.0, 12.0, 2.0, 3.0 };
    for (double side : testCases) {
    double[] sides = otherSides(side);
    double longest = sides[0];
    double medium = sides[1];
    System.out.printf("Shortest Side: %.1f, Longest Side: %.2f, Medium
Side: %.2f%n", side, longest, medium);
    }
```

```
}
}
// Shortest Side: 1.0, Longest Side: 2.00, Medium Side: 1.73
// Shortest Side: 12.0, Longest Side: 24.00, Medium Side: 20.78
// Shortest Side: 2.0, Longest Side: 4.00, Medium Side: 3.46
// Shortest Side: 3.0, Longest Side: 6.00, Medium Side: 5.20
```

194. WAR OF NUMBERS

```
i mport java.util.Arrays ;
    public class WarOfNumbers {
    // Method to calculate the absolute difference between the sum of even and odd numbers
    public static int warOfNumbers(int[] numbers) {
    int evenSum = 0;
    int oddSum = 0;
    for (int num : numbers) {
    if (num % 2 == 0) {
    evenSum += num; // Add to even sum if the number is even
    } else {
    oddSum += num; // Add to odd sum if the number is odd
    }
    }
    return Math.abs(evenSum - oddSum); // Return the absolute difference
    }
    public static void main(String[] args) {
    int[][] examples = {
    {2, 8, 7, 5},
    {12, 90, 75},
    {5, 9, 45, 6, 2, 7, 34, 8, 6, 90, 5, 243}
```

```
    };
    for (int[] numbers : examples) {
    System.out.println(warOfNumbers(numbers)); // Print the result for
each example
    }
    }
    }
    // 2
    // 27
    // 168
```

195. MAKE A CIRCLE WITH OOP

```
p ublic class Circle {
    private double radius;
    // Constructor to initialize the radius
    public Circle(double radius) {
    this.radius = radius;
    }
    // Method to calculate the area of the circle
    public double getArea() {
    return Math.PI * radius * radius;
    }
    // Method to calculate the perimeter of the circle
    public double getPerimeter() {
    return 2 * Math.PI * radius;
    }
    public static void main(String[] args) {
    double[] circles = {11.0, 4.44};
    for (double radius : circles) {
    Circle circle = new Circle(radius);
    System.out.printf("Radius: %.2f - Area: %.6f, Perimeter: %.6f%n",
    radius, circle.getArea(), circle.getPerimeter());
    }
```

```
}
}
// Radius: 11.00 - Area: 380.132711, Perimeter: 69.115038
// Radius: 4.44 - Area: 61.932101, Perimeter: 27.897343
```

196. FIND THE NTH TETRAHEDRAL NUMBER

```
p ublic class Tetrahedral {
    public static long tetra(long n) {
    return n * (n + 1) * (n + 2) / 6;
    }
    public static void main(String[] args) {
    System.out.println(tetra(2));  // Output: 4
    System.out.println(tetra(5));  // Output: 35
    System.out.println(tetra(6));  // Output: 56
    }
    }
```

197. JOKE TELLER

```
i mport java.util.Random ;
    import java.util.Scanner;
    public class JokeTeller {
    public static void jokeTellerProgram() {
    String[] categories = {"knock-knock", "dad", "animal", "puns"};
    Random rand = new Random();
    String category = categories[rand.nextInt(categories.length)];
    String setup = "";
    String punchline1 = "";
    String punchline2 = "";
    switch (category) {
    case "knock-knock":
    setup = "Knock, knock.";
    punchline1 = "Tank.";
    punchline2 = "You're welcome!";
    break;
    case "dad":
    setup = "Why did the scarecrow win an award?";
    punchline2 = "Because he was outstanding in his field!";
    break;
    case "animal":
```

```
setup = "Why don't scientists trust atoms?";
punchline2 = "Because they make up everything!";
break;
default: // for "puns"
setup = "I used to be a baker because I kneaded dough.";
break;
}
if (!setup.isEmpty()) {
System.out.println(setup);
}
if (!punchline1.isEmpty()) {
Scanner scanner = new Scanner(System.in);
System.out.print(punchline1 + " Press enter for the punchline...");
scanner.nextLine();
}
System.out.println(punchline2);
}
public static void main(String[] args) {
jokeTellerProgram();
}
}
// Why did the scarecrow win an award?
// Press enter for the punchline...
```

// Because he was outstanding in his field!

198. WHICH GENERATION ARE YOU?

```
p ublic class GenerationChecker {
    public static String generation(int x, char y) {
    switch (x) {
    case -3:
    return (y == 'm') ? "great grandfather" : "great grandmother";
    case -2:
    return (y == 'm') ? "grandfather" : "grandmother";
    case -1:
    return (y == 'm') ? "father" : "mother";
    case 0:
    return "me!";
    case 1:
    return (y == 'm') ? "son" : "daughter";
    case 2:
    return (y == 'm') ? "grandson" : "granddaughter";
    case 3:
    return (y == 'm') ? "great grandson" : "great granddaughter";
    default:
    return "unknown";
    }
    }
```

```
public static void main(String[] args) {
System.out.println(generation(2, 'f'));  // granddaughter
System.out.println(generation(-3, 'm')); // great grandfather
System.out.println(generation(1, 'f'));  // daughter
}
}
```

199. FIZZBUZZ GAME

```
p ublic class FizzBuzzGame {
    public static void fizzBuzzGame() {
    System.out.println("Welcome to the FizzBuzz Game!");
    System.out.println("Let's play FizzBuzz!");
    for (int i = 1; i <= 100; i++) {
    String output = "";
    if (i % 3 == 0) {
    output += "Fizz";
    }
    if (i % 5 == 0) {
    output += "Buzz";
    }
    // Print the output or the number itself
    if (!output.isEmpty()) {
    System.out.println(output);
    } else {
    System.out.println(i);
    }
    }
    }
    public static void main(String[] args) {
```

```
fizzBuzzGame();
}
}
// Welcome to the FizzBuzz Game!
// Let's play FizzBuzz!
// 1
// 2
// Fizz
// 4
// Buzz
// Fizz
// 7
// 8
// Fizz
// Buzz
// ...
// FizzBuzz
```

200. SWAP PAIRS OF ADJACENT DIGITS

```
i mport java.util.Scanner ;
    public class SwapPairsOfAdjacentDigits {
    public static void swapPairsOfAdjacentDigits(int number) {
    String numberStr = String.valueOf(number);
    int length = numberStr.length();
    // Check if the number has an even length
    if (length % 2 != 0) {
    System.out.println("Please enter an integer with an even length.");
    return;
    }
    // Swap pairs of adjacent digits
    StringBuilder swappedNumber = new StringBuilder();
    for (int i = 0; i < length; i += 2) {
    swappedNumber.append(numberStr.charAt(i + 1)); // Add the next digit first
    swappedNumber.append(numberStr.charAt(i));     // Then add the current digit
    }
    // Convert the result back to an integer
    int result = Integer.parseInt(swappedNumber.toString());
    System.out.println("Original Number: " + number);
```

```
    System.out.println("Number with Swapped Pairs of Adjacent Digits: " +
result);
    }
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number with an even number of digits: ");
    int number = scanner.nextInt();
    swapPairsOfAdjacentDigits(number);
    scanner.close();
    }
    }
    // Enter a number with an even number of digits: 123456
    // Original Number: 123456
    // Number with Swapped Pairs of Adjacent Digits: 214365
```

201. CAPITALIZATION CHANGER

```
p ublic class CapitalizationChanger {
    public static String changeCapitalization(String inputString) {
    StringBuilder resultString = new StringBuilder();
    for (char ch : inputString.toCharArray()) {
    // Check if the character is uppercase
    if (Character.isUpperCase(ch)) {
    // Convert uppercase to lowercase
    resultString.append(Character.toLowerCase(ch));
    } else {
    // Convert lowercase to uppercase
    resultString.append(Character.toUpperCase(ch));
    }
    }
    System.out.println("Original String: " + inputString);
    System.out.println("String with Changed Capitalization: " +
resultString);
    return resultString.toString();
    }
    public static void main(String[] args) {
    String input = "Hello World";
    changeCapitalization(input);
```

```
}
}
// Original String: Hello World
// String with Changed Capitalization: hELLO wORLD
```

202. ARRAY HALVES SWAPPER

```
i mport java.util.Arrays ;
    public class ArrayHalvesSwapper {
    public static void swapArrayHalves(int[] arr) {
    int length = arr.length;
    // Check if the array has an even length
    if (length % 2 != 0) {
    System.out.println("Please provide an array with an even length.");
    return;
    }
    // Calculate the midpoint of the array
    int midpoint = length / 2;
    // Swap the two halves of the array
    for (int i = 0; i < midpoint; i++) {
    // Swap elements
    int temp = arr[i];
    arr[i] = arr[midpoint + i];
    arr[midpoint + i] = temp;
    }
    System.out.println("Array with Swapped Halves: " +
Arrays.toString(arr));
    }
```

```
public static void main(String[] args) {
int[] array = {1, 2, 3, 4, 5, 6};
swapArrayHalves(array);
}
}
// Array with Swapped Halves: [4, 5, 6, 1, 2, 3]
```

203. SUM OF DIGITS IN STRING

```
p ublic class SumOfDigitsInString {
    public static void sumOfDigitsInString(String inputString) {
    int digitSum = 0;
    // Iterate over each character in the string
    for (char ch : inputString.toCharArray()) {
    // Check if the character is a digit
    if (Character.isDigit(ch)) {
    // Convert the character to its numerical value and add to sum
    digitSum += Character.getNumericValue(ch);
    }
    }
    System.out.println("Original String: " + inputString);
    System.out.println("Sum of Digits in the String: " + digitSum);
    }
    public static void main(String[] args) {
    sumOfDigitsInString("abc123xyz456");
    }
    }
    // Original String: abc123xyz456
    // Sum of Digits in the String: 21
```

204. SUM OF CUBES

```
p ublic class SumOfCubes {
    public static void sumOfCubes(int upToInteger) {
    int cubesSum = 0;
    // Calculate the sum of cubes from 1 to upToInteger
    for (int i = 1; i <= upToInteger; i++) {
    cubesSum += Math.pow(i, 3); // Calculate the cube of each integer and add to sum
    }
    System.out.println("Sum of Cubes from 1 to " + upToInteger + ": " + cubesSum);
    }
    public static void main(String[] args) {
    sumOfCubes(5);
    }
    }
    // Sum of Cubes from 1 to 5: 225
```

205. MAXIMUM INTEGER FOR SUM

```
p ublic class MaximumIntegerForSum {
    public static void findMaxIntegerForSum(int targetSum) {
    int currentSum = 0;
    int maxInteger = 0;
    // Find the maximum integer (n) for which the sum is <= targetSum
    while (currentSum + maxInteger + 1 <= targetSum) {
    maxInteger++;
    currentSum += maxInteger;
    }
    System.out.println("Maximum Integer (n) for Sum <= " + targetSum +
": " + maxInteger);
    }
    public static void main(String[] args) {
    findMaxIntegerForSum(15);
    }
    }
    // Maximum Integer (n) for Sum <= 15: 5
```

206. URL BREAKDOWN

```
i mport java.util.HashMap ;
    import java.util.regex.Matcher;
    import java.util.regex.Pattern;
    public class URLBreakdown {
    public static void breakUrl(String url) {
    // Regular expression to match the URL components
    String urlRegex = "^(\\w+):\\/\\/(\\w+([\\.\\w-]*)?)(\\/.*)?$";
    Pattern pattern = Pattern.compile(urlRegex);
    Matcher matcher = pattern.matcher(url);
    // HashMap to store the URL parts
    HashMap<String, String> urlParts = new HashMap<>();
    if (matcher.find()) {
    urlParts.put("scheme", matcher.group(1));
    urlParts.put("domain", matcher.group(2));
    urlParts.put("path", matcher.group(4) != null ? matcher.group(4) : "");
    } else {
    System.out.println("Invalid URL format.");
    return;
    }
    // Print URL parts
    System.out.println("URL Parts:");
```

```
for (HashMap.Entry<String, String> entry : urlParts.entrySet()) {
System.out.println(entry.getKey() + ": " + entry.getValue());
}
}
public static void main(String[] args) {
breakUrl("https://www.example.org/page");
}
}
// URL Parts:
// scheme: https
// domain: www.example.org
// path: /page
```

207. SORT STRINGS BY LENGTH

```
i mport java.util.Arrays ;
    public class SortStringsByLength {
    public static void sortStringsByLength(String[] stringsArray) {
    // Copy the original array to maintain its state
    String[] sortedArray = Arrays.copyOf(stringsArray,
stringsArray.length);
    // Sort the array by length
    Arrays.sort(sortedArray, (s1, s2) -> Integer.compare(s1.length(),
s2.length()));
    // Print the original and sorted arrays
    System.out.println("Original Array of Strings:");
    System.out.println(Arrays.toString(stringsArray));
    System.out.println("Array of Strings Sorted by Length:");
    System.out.println(Arrays.toString(sortedArray));
    }
    public static void main(String[] args) {
    sortStringsByLength(new String[]{"apple", "banana", "orange", "kiwi",
"grape"});
    }
    }
    // Original Array of Strings:
    // [apple, banana, orange, kiwi, grape]
```

```
// Array of Strings Sorted by Length:
// [kiwi, apple, grape, banana, orange]
```

208. SIMPLIFY ABSOLUTE PATH

```
i mport java.util.ArrayList ;
    import java.util.List;
    public class SimplifyAbsolutePath {
    public static void simplifyAbsolutePath(String path) {
    String[] parts = path.split("/");
    List<String> simplifiedParts = new ArrayList<>();
    for (String part : parts) {
    switch (part) {
    case "..":
    if (!simplifiedParts.isEmpty()) {
    simplifiedParts.remove(simplifiedParts.size() - 1); // Move up one level
for '..'
    }
    break;
    case "":
    case ".":
    // Ignore empty parts and current directory '.'
    break;
    default:
    simplifiedParts.add(part);
    break;
```

```
}
}
String simplifiedPath = "/" + String.join("/", simplifiedParts);
System.out.println("Original Absolute Path: " + path);
System.out.println("Simplified Absolute Path: " + simplifiedPath);
}
public static void main(String[] args) {
simplifyAbsolutePath("/home/user/../documents/./file.txt");
}
}
// Original Absolute Path: /home/user/../documents/./file.txt
// Simplified Absolute Path: /home/documents/file.txt
```

209. COUNT COMMON ELEMENTS IN ARRAYS

```
i mport java.util.Arrays ;
    import java.util.HashSet;
    import java.util.Set;
    public class CountCommonElements {
    public static void countCommonElements(int[] arr1, int[] arr2) {
    Set<Integer> set1 = new HashSet<>();
    Set<Integer> set2 = new HashSet<>();
    // Add elements of the first array to the set
    for (int num : arr1) {
    set1.add(num);
    }
    // Add elements of the second array to the set
    for (int num : arr2) {
    set2.add(num);
    }
    // Find common elements using intersection
    set1.retainAll(set2);
    // Convert the set back to an array (optional)
    Integer[] commonElements = set1.toArray(new Integer[0]);
    int numberOfCommonElements = commonElements.length;
```

```
    // Print results
    System.out.println("Array 1: " + Arrays.toString(arr1));
    System.out.println("Array 2: " + Arrays.toString(arr2));
    System.out.println("Common Elements: " +
Arrays.toString(commonElements));
    System.out.println("Number of Common Elements: " +
numberOfCommonElements);
    }
    public static void main(String[] args) {
    countCommonElements(new int[]{1, 2, 3, 4, 5}, new int[]{3, 4, 5, 6,
7});
    }
    }
    // Array 1: [1, 2, 3, 4, 5]
    // Array 2: [3, 4, 5, 6, 7]
    // Common Elements: [3, 4, 5]
    // Number of Common Elements: 3
```

210. CHECK SAME DIGITS IN A NUMBER

```
p ublic class CheckSameDigits {
    public static void areAllDigitsSame(int number) {
    String numberStr = Integer.toString(number);
    char firstDigit = numberStr.charAt(0);
    // Check if all digits are the same
    for (int i = 1; i < numberStr.length(); i++) {
    if (numberStr.charAt(i) != firstDigit) {
    System.out.println("Digits in " + number + " are not all the same.");
    return;
    }
    }
    System.out.println("Digits in " + number + " are all the same.");
    }
    public static void main(String[] args) {
    areAllDigitsSame(22222);
    }
    }
    // Digits in 22222 are all the same.
```

211. RIGHTMOST ROUND NUMBER POSITION

```
p ublic class RightmostRoundNumberPosition {
    public static void rightmostRoundNumberPosition(int[] arr) {
    for (int i = arr.length - 1; i >= 0; i—) {
    if (arr[i] % 10 == 0) {
    System.out.println("Rightmost Round Number: " + arr[i] + ", Position: "
+ (i + 1));
    return;
    }
    }
    System.out.println("No round numbers found in the array. Position: 0");
    }
    public static void main(String[] args) {
    rightmostRoundNumberPosition(new int[]{123, 450, 678, 900});
    }
    }
    // Rightmost Round Number: 900, Position: 4
```

212. REVERSE BITS OF 16-BIT UNSIGNED SHORT INTEGER

```
p ublic class ReverseBits {
    public static void reverseBits16BitUnsignedShort(int integer) {
    if (integer < 0 || integer > 65535) {
    System.out.println("Please provide a 16-bit unsigned short integer.");
    return;
    }
    // Get the binary representation of the integer
    String binaryRepresentation = String.format("%16s",
Integer.toBinaryString(integer)).replace(' ', '0');
    // Reverse the binary representation
    String reversedBinary = new
StringBuilder(binaryRepresentation).reverse().toString();
    // Convert reversed binary back to an integer
    int reversedInteger = Integer.parseUnsignedInt(reversedBinary, 2);
    System.out.println("Original Integer: " + integer);
    System.out.println("Binary Representation: " + binaryRepresentation);
    System.out.println("Reversed Binary: " + reversedBinary);
    System.out.println("Reversed Integer: " + reversedInteger);
    }
    public static void main(String[] args) {
    reverseBits16BitUnsignedShort(5678);
```

```
}
}
// Original Integer: 5678
// Binary Representation: 0001011000101110
// Reversed Binary: 0111010001101000
// Reversed Integer: 29800
```

213. GREATER THAN 15 CHECKER

```
p ublic class GreaterThan15Checker {
    public static void greaterThan15Checker(int number) {
    int result = number > 15 ? number : 15;
    System.out.println("Given Number: " + number);
    System.out.println("Result: " + result);
    }
    public static void main(String[] args) {
    greaterThan15Checker(20);
    }
    }
    // Given Number: 20
    // Result: 20
```

214. REPLACE FIRST DIGIT WITH $

```
p ublic class ReplaceFirstDigitWithDollar {
    public static void replaceFirstDigitWithDollar(String inputString) {
    boolean foundDigit = false;
    StringBuilder modifiedString = new StringBuilder();
    for (char c : inputString.toCharArray()) {
    if (Character.isDigit(c) && !foundDigit) {
    modifiedString.append('$');
    foundDigit = true; // Mark that we found the first digit
    } else {
    modifiedString.append(c);
    }
    }
    System.out.println("Original String: " + inputString);
    System.out.println("Modified String: " + modifiedString.toString());
    }
    public static void main(String[] args) {
    replaceFirstDigitWithDollar("abc123xyz456");
    }
    }
    // Original String: abc123xyz456
    // Modified String: abc$23xyz456
```

215. PREFIX SUMS

```
i mport java.util.Arrays ;
    public class PrefixSums {
    public static void prefixSums(int[] inputArray) {
    int prefixSum = 0;
    int[] prefixSumArray = new int[inputArray.length];
    for (int i = 0; i < inputArray.length; i++) {
    prefixSum += inputArray[i];
    prefixSumArray[i] = prefixSum;
    }
    System.out.println("Original Array: " + Arrays.toString(inputArray));
    System.out.println("Prefix Sums Array: " +
Arrays.toString(prefixSumArray));
    }
    public static void main(String[] args) {
    prefixSums(new int[]{1, 2, 3, 4, 5});
    }
    }
    // Original Array: [1, 2, 3, 4, 5]
    // Prefix Sums Array: [1, 3, 6, 10, 15]
```

216. NEXT PRIME NUMBER

```
p ublic class NextPrimeNumber {
    // Function to check if a number is prime
    public static boolean isPrime(int num) {
    if (num < 2) {
    return false;
    }
    for (int i = 2; i <= Math.sqrt(num); i++) {
    if (num % i == 0) {
    return false;
    }
    }
    return true;
    }
    // Function to find the next prime number
    public static int nextPrimeNumber(int givenNumber) {
    int nextNumber = givenNumber + 1;
    while (!isPrime(nextNumber)) {
    nextNumber++;
    }
    return nextNumber;
    }
```

```
public static void main(String[] args) {
int givenNumber = 10;
int nextPrime = nextPrimeNumber(givenNumber);
System.out.println("Given Number: " + givenNumber);
System.out.println("Next Prime Number: " + nextPrime);
}
}
// Given Number: 10
// Next Prime Number: 11
```

217. REVERSE ORDER OF BITS

```
p ublic class ReverseOrderOfBits {
    // Function to reverse the order of bits in a given byte
    public static void reverseOrderOfBits(int integer) {
    String binaryRepresentation = String.format("%08d",
Integer.parseInt(Integer.toBinaryString(integer)));
    String reversedBinary = new
StringBuilder(binaryRepresentation).reverse().toString();
    int reversedInteger = Integer.parseInt(reversedBinary, 2);
    System.out.println("Original Integer: " + integer);
    System.out.println("Binary Representation: " + binaryRepresentation);
    System.out.println("Reversed Binary: " + reversedBinary);
    System.out.println("Reversed Integer: " + reversedInteger);
    }
    public static void main(String[] args) {
    reverseOrderOfBits(14);
    reverseOrderOfBits(56);
    reverseOrderOfBits(234);
    }
    }
    // Original Integer: 14
    // Binary Representation: 00001110
```

// Reversed Binary: 01110000

// Reversed Integer: 112

// Original Integer: 56

// Binary Representation: 00111000

// Reversed Binary: 00011100

// Reversed Integer: 28

// Original Integer: 234

// Binary Representation: 11101010

// Reversed Binary: 01010111

// Reversed Integer: 87

218. PYRAMID PATTERN

```
p ublic class PyramidPattern {
    // Function to generate an ASCII triangle (pyramid pattern)
    public static void generateAsciiTriangle(int height) {
    for (int i = 1; i <= height; i++) {
    String spaces = " ".repeat(height - i);
    String stars = "*".repeat(i * 2 - 1);
    System.out.println(spaces + stars);
    }
    }
    public static void main(String[] args) {
    int triangleHeight = 5;
    generateAsciiTriangle(triangleHeight);
    }
    }
    //     *
    //    ***
    //   *****
    //  *******
    // *********
```

CONGRATULATIONS!

I n very **line of code** , they have woven a story of innovation and creativity.

This book has been your compass in the vast world of **Java** .

Close this chapter knowing that every challenge overcome is an achievement, and every solution is a step toward mastery.

Thank you for allowing me to be part of your journey.

With gratitude, **Hernando Abella** ...

Author of **200+ Java Programs for Beginners**

Discover other useful resources:

www.beat-byte-publishing.com

Get your bonus books here:

www.hernandoabella.com

A LUNA PUBLISHING HOUSE