Java Coding Interview Questions

1. Write a Java program to check if a number is prime.

```
public class PrimeCheck {
    public static void main(String[] args) {
        int num = 29;
        boolean isPrime = true;

        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                isPrime = false;
                break;
            }
        }

        if (isPrime)
            System.out.println(num + " is a prime number.");
        else
            System.out.println(num + " is not a prime
number.");
    }
}
```

Output: 29 is a prime number.

2. Write a Java program to sort an array using the bubble sort algorithm.

```
import java.util.Arrays;

public class BubbleSort {
    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        bubbleSort(arr);
        System.out.println("Sorted array: " +
Arrays.toString(arr));
    }
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    // swap arr[j] and arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
}
```

Output: Sorted array: [11, 12, 22, 25, 34, 64, 90]

3. Write a Java program to reverse a string.

```
public class ReverseString {
    public static void main(String[] args) {
        String str = "Java";
        String reversed = new
StringBuilder(str).reverse().toString();
        System.out.println("Reversed string: " + reversed);
    }
}
```

Output: Reversed string: avaJ

4. Write a Java program to check if a string is a rotation of another string.

```
public class StringRotation {
    public static void main(String[] args) {
        String str1 = "abcd";
        String str2 = "cdab";
        boolean isRotation = areRotations(str1, str2);
        System.out.println(str1 + " and " + str2 + " are
rotations: " + isRotation);
    }
    public static boolean areRotations(String str1, String
str2) {
        if (str1.length() != str2.length()) return false;
        String temp = str1 + str1;
        return temp.contains(str2);
    }
}
```

Output: abcd and cdab are rotations: true

5. Write a Java program to check for balanced parentheses in an expression.

```
import java.util.Stack;

public class BalancedParentheses {
    public static void main(String[] args) {
        String expression = "{[()]}”;
        boolean isBalanced = checkBalanced(expression);
        System.out.println("Is the expression balanced? " +
isBalanced;
    }

    public static boolean checkBalanced(String expr) {
        Stack<Character> stack = new Stack<>();
        for (char ch: expr.toCharArray()) {
            if (ch == '{' || ch == '(' || ch == '[') {
                stack.push(ch);
            } else if (ch == '}' || ch == ')' || ch == ']') {
                if (stack.isEmpty()) return false;
                char top = stack.pop();
                if (!isMatchingPair(top, ch)) return false;
            }
        }
        return stack.isEmpty();
    }

    public static boolean isMatchingPair(char open, char close)
{
        return (open == '{' && close == '}') ||
               (open == '(' && close == ')') ||
               (open == '[' && close == ']');
    }
}
```

Output: Is the expression balanced? true

6. Write a Java program to find the largest number in an array.

```
public class LargestNumber {
    public static void main(String[] args) {
        int[] arr = {10, 20, 5, 40, 25};
        int max = arr[0];

        for (int num : arr) {
            if (num > max) {
                max = num;
            }
        }

        System.out.println("The largest number is: " + max);
    }
}
```

Output: The largest number is: 40

7. Write a Java program to print the Fibonacci series up to n terms.

```
public class Fibonacci {
    public static void main(String[] args) {
        int n = 10, a = 0, b = 1;
        System.out.print("Fibonacci Series: " + a + " " + b);

        for (int i = 2; i < n; i++) {
            int next = a + b;
            System.out.print(" " + next);
            a = b;
            b = next;
        }
    }
}
```

Output: Fibonacci Series: 0 1 1 2 3 5 8 13 21 34

8. Write a Java program to find the sum of digits of a number.

```
public class SumOfDigits {
    public static void main(String[] args) {
        int num = 1234, sum = 0;

        while (num != 0) {
            sum += num % 10;
            num /= 10;
        }

        System.out.println("Sum of digits: " + sum);
    }
}
```

Output: Sum of digits: 10

9. Write a Java program to print all prime numbers up to n.

```
public class PrimeNumbersUpToN {
    public static void main(String[] args) {
        int n = 20;
        System.out.print("Prime numbers up to " + n + ": ");

        for (int i = 2; i <= n; i++) {
            if (isPrime(i)) {
                System.out.print(i + " ");
            }
        }
    }

    public static boolean isPrime(int num) {
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

Output: Prime numbers up to 20: 2 3 5 7 11 13 17 19

10. Write a Java program to find the length of the longest substring without repeating characters.

```
import java.util.HashSet;

public class LongestSubstring {
    public static void main(String[] args) {
        String str = "abcabcbb";
        int length = lengthOfLongestSubstring(str);
        System.out.println("Length of longest substring: " +
length);
    }
    public static int lengthOfLongestSubstring(String s) {
        HashSet<Character> set = new HashSet<>();
        int maxLength = 0, left = 0;
        for (int right = 0; right < s.length(); right++) {
            while (set.contains(s.charAt(right))) {
                set.remove(s.charAt(left++));
            }
            set.add(s.charAt(right));
            maxLength = Math.max(maxLength, right - left + 1);
        }
        return maxLength;
    }
}
```

Output: Length of longest substring: 3

11. Write a Java program to count the number of vowels in a string.

```
public class CountVowels {
    public static void main(String[] args) {
        String str = "Hello World";
        int count = countVowels(str);
        System.out.println("Number of vowels: " + count);
    }

    public static int countVowels(String str) {
        int count = 0;
        for (char c : str.toLowerCase().toCharArray()) {
            if (c == 'a' || c == 'e' || c == 'i' || c == 'o' ||
c == 'u') {
                count++;
            }
        }
        return count;
    }
}
```

Output: Number of vowels: 3

12. Write a Java program to implement a simple banking system (deposit and withdrawal).

```
import java.util.Scanner;

public class SimpleBanking {
    private double balance;

  public SimpleBanking() {
        this.balance = 0.0;
    }

  public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount);
    }

  public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
 public double getBalance() {
     return balance;
    }
 public static void main(String[] args) {
  Scanner scanner = new Scanner(System.in);
  SimpleBanking account = new SimpleBanking();

while (true)
{
 System.out.println("\n1. Deposit\n2. Withdraw\n3.Check 
Balance\n4. Exit");
 System.out.print("Choose an option: ");
 int choice = scanner.nextInt();
 switch (choice)
{
case 1:
 System.out.print("Enter amount to deposit: ");
 double depositAmount = scanner.nextDouble();
 account.deposit(depositAmount);
     Break;

case 2:
 System.out.print("Enter amount to withdraw: ");
 double withdrawAmount = scanner.nextDouble();
 account.withdraw(withdrawAmount);
     Break;

Case 3:
 System.out.println("Current balance: " + 
account.getBalance());
     Break;

Case 4:
 System.out.println("Exiting...");
 scanner.close();
 return;
 Default:
 System.out.println("Invalid choice. Try again.");
            }
        }
    }
}
```

Output:

```
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option: 1
Enter the amount to deposit: 100
Deposited: 100.0
```

13. Write a Java program to implement bubble sort.

```
import java.util.Arrays;

public class BubbleSort {
    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        bubbleSort(arr);
        System.out.println("Sorted array: " +
Arrays.toString(arr));
    }

    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    // swap arr[j] and arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
}
```

Output: Sorted array: [11, 12, 22, 25, 34, 64, 90]

14. Write a Java program to count the occurrences of each character in a string.

```
import java.util.HashMap;
public class CharCount {
    public static void main(String[] args) {
        String str = "java";
        HashMap<Character, Integer> charCount = new
HashMap<>();
        for (char c: str.toCharArray()) {
            charCount.put(c, charCount.getOrDefault(c, 0) + 1);
        }
        System.out.println("Character count: " + charCount);
    }
}
```

Output: Character count: {a=2, v=1, j=1}

15. Write a Java program to find the length of the longest palindrome in a string.

```
public class LongestPalindrome {
    public static void main(String[] args) {
        String str = "babad";
        System.out.println("Longest palindrome: " +
longestPalindrome(str));
    }

    public static String longestPalindrome(String s) {
        int maxLength = 1, start = 0, len = s.length();

        for (int i = 0; i < len; i++) {
            for (int j = i; j < len; j++) {
                int flag = 1;

                for (int k = 0; k < (j - i + 1) / 2; k++)
                    if (s.charAt(i + k) != s.charAt(j - k))
                        flag = 0;

                if (flag != 0 && (j - i + 1) > maxLength) {
                    start = i;
                    maxLength = j - i + 1;
                }
            }
        }

        return s.substring(start, start + maxLength);
    }
}
```

Output: Longest palindrome: bab

16. Write a Java program to find the maximum element in an array.

```
public class MaxElement {
    public static void main(String[] args) {
        int[] arr = {10, 20, 4, 45, 99};
        int max = findMax(arr);
        System.out.println("Maximum element: " + max);
    }

    public static int findMax(int[] arr) {
        int max = arr[0];
        for (int num: arr) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }
}
```

Output: Maximum element: 99

17. Write a Java program to calculate the sum of all elements in a 2D array.

```
public class Sum2DArray {
    public static void main(String[] args) {
        int[][] arr = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        int sum = 0;

        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr[i].length; j++) {
                sum += arr[i][j];
            }
        }
        System.out.println("Sum of all elements: " + sum);
    }
}
```

Output: Sum of all elements: 45

18. Write a Java program to find the transpose of a matrix.

```
public class TransposeMatrix {
    public static void main(String[] args) {
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        int[][] transpose = new
int[matrix[0].length][matrix.length];

        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                transpose[j][i] = matrix[i][j];
            }
        }

        System.out.println("Transpose of the matrix:");
        for (int i = 0; i < transpose.length; i++) {
            for (int j = 0; j < transpose[0].length; j++) {
                System.out.print(transpose[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output: Transpose of the matrix:

```
1 4 7
2 5 8
3 6 9
```

19. Write a Java program to convert a binary number to decimal.

```
public class BinaryToDecimal {
    public static void main(String[] args) {
        String binary = "1010";
        int decimal = Integer.parseInt(binary, 2);
        System.out.println("Decimal of " + binary + " is: " +
decimal);
    }
}
```

Output: The decimal of 1010 is: 10

20. Write a Java program to find the power of a number using recursion.

```
public class PowerRecursion {
    public static void main(String[] args) {
        int base = 2, exp = 5;
        System.out.println(base + "^" + exp + " = " +
power(base, exp));
    }

    public static int power(int base, int exp) {
        if (exp == 0)
            return 1;
        return base * power(base, exp - 1);
    }
}
```

Output: 2^5 = 32

21. Write a Java program to implement a simple calculator (addition, subtraction, multiplication, division).

```
import java.util.Scanner;

public class SimpleCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first number: ");
        double num1 = scanner.nextDouble();
        System.out.print("Enter second number: ");
        double num2 = scanner.nextDouble();
        System.out.print("Choose operation (+, -, *, /): ");
        char operation = scanner.next().charAt(0);

        double result;
        switch (operation) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                result = num1 / num2;
                break;
            default:
                System.out.println("Invalid operation");
                return;
        }
        System.out.println("Result: " + result);
        scanner.close();
    }
}
```

Output:

```
Enter the first number: 10
Enter the second number: 5
Choose operation (+, -, *, /): +
Result: 15.0
```

22. What is the output of the following code?

```
public class Main {
   public static void main(String[] args) {
       String str1 = "Hello";
       String str2 = "Hello";
       String str3 = new String("Hello");

       System.out.println(str1 == str2);
       System.out.println(str1 == str3);
       System.out.println(str1.equals(str3));
   }
}
```

Output:

```
True
False
True
```

23. Write a Java program to find the intersection of two arrays.

```java
import java.util.HashSet;

public class IntersectionArrays {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {4, 5, 6, 7, 8};

        HashSet<Integer> set = new HashSet<>();
        for (int i: arr1) {
            set.add(i);
        }

        System.out.print("Intersection: ");
        for (int i: arr2) {
            if (set.contains(i)) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Output: Intersection: 4 5

24. Write a Java program to find an array's maximum and minimum number.

```java
public class MaxMinInArray {
    public static void main(String[] args) {
        int[] arr = {10, 20, 5, 25, 15};
        int max = arr[0], min = arr[0];

        for (int num : arr) {
            if (num > max) {
                max = num;
            }
            if (num < min) {
                min = num;
            }
        }

        System.out.println("Maximum number is: " + max);
        System.out.println("Minimum number is: " + min);
    }
}
```

Output:

Maximum number is: 25

Minimum number is: 5

25. Write a Java program to find the sum of the first n natural numbers.

```
public class SumNaturalNumbers {
    public static void main(String[] args) {
        int n = 10;
        int sum = n * (n + 1) / 2;
        System.out.println("Sum of first " + n + " natural
numbers: " + sum);
    }
}
```

Output: Sum of first 10 natural numbers: 55

26. Write a Java program to find the common elements between two arrays.

```
import java.util.HashSet;

public class CommonElements {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {3, 4, 5, 6, 7};

        HashSet<Integer> set = new HashSet<>();
        for (int i: arr1) {
            set.add(i);
        }

        System.out.print("Common elements: ");
        for (int i: arr2) {
            if (set.contains(i)) {
                System.out.print(i + " ");
                }
            }
        }
}
```

Output: Common elements: 3 4 5

27. Write a Java program to find the largest element in each row of a 2D array.

```
public class LargestInRow {
    public static void main(String[] args) {
        int[][] arr = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

        for (int i = 0; i < arr.length; i++) {
            int max = arr[i][0];
            for (int j = 1; j < arr[i].length; j++) {
                if (arr[i][j] > max) {
                    max = arr[i][j];
                }
            }
            System.out.println("Largest element in row " + (i +
1) + ": " + max);
        }
    }
}
```

Output:

```
Largest element in row 1: 3
Largest element in row 2: 6
Largest element in row
```

28. Write a Java program to remove duplicates from an array.

```
import java.util.Arrays;
import java.util.LinkedHashSet;

public class RemoveDuplicates {
    public static void main(String[] args) {
        Integer[] arr = {1, 2, 2, 3, 4, 4, 5};
        LinkedHashSet<Integer> set = new
LinkedHashSet<>(Arrays.asList(arr));

        Integer[] newArr = set.toArray(new Integer[0]);
        System.out.println("Array after removing duplicates: "
+ Arrays.toString(newArr));
    }
}
```

Output: Array after removing duplicates: [1, 2, 3, 4, 5]

29. Write a Java program to convert a decimal number to binary.

```
public class DecimalToBinary {
    public static void main(String[] args) {
        int decimal = 10;
        String binary = Integer.toBinaryString(decimal);
        System.out.println("Binary of " + decimal + " is: " +
binary);
    }
}
```

Output: Binary of 10 is: 1010

30. Write a Java program to check if a number is a power of two.

```
public class PowerOfTwo {
    public static void main(String[] args) {
        int num = 16;
        boolean result = isPowerOfTwo(num);
        if (result) {
            System.out.println(num + " is a power of two.");
        } else {
            System.out.println(num + " is not a power of
two.");
        }
    }

    public static boolean isPowerOfTwo(int num) {
        if (num <= 0) {
            return false;
        }
        return (num & (num - 1)) == 0;
    }
}
```

Output: 16 is a power of two.

31. Write a Java program to implement binary search.

```
import java.util.Arrays;

public class BinarySearch {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40, 50};
        int target = 30;
        Arrays.sort(arr);  // Binary search requires sorted
array
        int index = binarySearch(arr, target, 0, arr.length -
1);
        if (index == -1) {
            System.out.println("Element not found.");
        } else {
            System.out.println("Element found at index: " +
index);
        }
    }

    public static int binarySearch(int[] arr, int target, int
low, int high) {
        if (low > high) {
            return -1;
        }
        int mid = (low + high) / 2;

        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] > target) {
            return binarySearch(arr, target, low, mid - 1);
        } else {
            return binarySearch(arr, target, mid + 1, high);
        }
    }
}
```

Output: Element found at index: 2

32. Write a Java program to swap two numbers without using a temporary variable.

```
public class SwapNumbers {
    public static void main(String[] args) {
        int a = 10, b = 20;

        System.out.println("Before swapping: a = " + a + ", b =
" + b);
        a = a + b;
        b = a - b;
        a = a - b;
        System.out.println("After swapping: a = " + a + ", b =
" + b);
    }
}
```

Output:

```
Before swapping: a = 10, b = 20
After swapping: a = 20, b = 10
```

33. Write a Java program to find the second largest number in an array.

```
public class SecondLargest {
    public static void main(String[] args) {
        int[] arr = {10, 20, 4, 45, 99};
        int first = Integer.MIN_VALUE, second =
Integer.MIN_VALUE;

        for (int num: arr) {
            if (num > first) {
                second = first;
                first = num;
            } else if (num > second && num != first) {
                second = num;
            }
        }

        if (second == Integer.MIN_VALUE) {
            System.out.println("No second largest element.");
        } else {
            System.out.println("Second largest number: " +
second);
        }
    }
}
```

Output: Second largest number: 45

34. Write a Java program to find the factorial of a number using recursion.

```
public class FactorialRecursion {
    public static void main(String[] args) {
        int num = 5;
        System.out.println("Factorial of " + num + " is: " +
factorial(num));
    }

    public static int factorial(int num) {
        if (num == 0) {
            return 1;
        }
        return num * factorial(num - 1);
    }
}
```

Output: Factorial of 5 is: 120

35. Write a Java program to check if a number is a perfect square.

```
public class PerfectSquare {
    public static void main(String[] args) {
        int num = 16;
        if (isPerfectSquare(num)) {
            System.out.println(num + " is a perfect square.");
        } else {
            System.out.println(num + " is not a perfect
square.");
        }
    }
```

```java

    public static boolean isPerfectSquare(int num) {
        int sqrt = (int) Math.sqrt(num);
        return sqrt * sqrt == num;
    }
}
```

Output: 16 is a perfect square.

36. Write a Java program to implement selection sort.

```java
import java.util.Arrays;

public class SelectionSort {
    public static void main(String[] args) {
        int[] arr = {29, 10, 14, 37, 13};
        selectionSort(arr);
        System.out.println("Sorted array: " +
Arrays.toString(arr));
    }

    public static void selectionSort(int[] arr) {
        for (int i = 0; i < arr.length - 1; i++) {
            int minIdx = i;
            for (int j = i + 1; j < arr.length; j++) {
                if (arr[j] < arr[minIdx]) {
                    minIdx = j;
                }
            }
            int temp = arr[minIdx];
            arr[minIdx] = arr[i];
            arr[i] = temp;
        }
    }
}
```

Output: Sorted array: [10, 13, 14, 29, 37]

37. Write a Java program to check if two strings are anagrams.

```
import java.util.Arrays;

public class AnagramCheck {
    public static void main(String[] args) {
        String str1 = "listen";
        String str2 = "silent";

        if (areAnagrams(str1, str2)) {
            System.out.println(str1 + " and " + str2 + " are
anagrams.");
        } else {
            System.out.println(str1 + " and " + str2 + " are
not anagrams.");
        }
    }

    public static boolean areAnagrams(String str1, String str2)
{
        if (str1.length() != str2.length()) {
            return false;
        }
        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        return Arrays.equals(arr1, arr2);
    }
}
```

Output: listen and silent are anagrams.

38. Write a Java program to find the missing number in an array of consecutive numbers.

```
public class MissingNumber {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 5, 6};
        int n = arr.length + 1;
        int totalSum = n * (n + 1) / 2;
        int arrSum = 0;
        for (int num : arr) {
            arrSum += num;
        }

        int missingNumber = totalSum - arrSum;
        System.out.println("The missing number is: " +
missingNumber);
    }
}
```

Output: The missing number is: 4

39. Write a Java program to check if a string is a palindrome.

```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "madam";
        if (isPalindrome(str)) {
            System.out.println(str + " is a palindrome.");
        } else {
            System.out.println(str + " is not a palindrome.");
        }
    }

    public static boolean isPalindrome(String str) {
        int left = 0;
        int right = str.length() - 1;

        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

Output: madam is a palindrome.

40. Write a Java program to merge two sorted arrays.

```java
import java.util.Arrays;

public class MergeSortedArrays {
    public static void main(String[] args) {
        int[] arr1 = {1, 3, 5, 7};
        int[] arr2 = {2, 4, 6, 8};
        int[] mergedArr = mergeArrays(arr1, arr2);
        System.out.println("Merged array: " +
Arrays.toString(mergedArr));
    }

    public static int[] mergeArrays(int[] arr1, int[] arr2) {
        int[] mergedArr = new int[arr1.length + arr2.length];
        int i = 0, j = 0, k = 0;

        while (i < arr1.length && j < arr2.length) {
            if (arr1[i] < arr2[j]) {
                mergedArr[k++] = arr1[i++];
            } else {
                mergedArr[k++] = arr2[j++];
            }
        }

        while (i < arr1.length) {
            mergedArr[k++] = arr1[i++];
        }

        while (j < arr2.length) {
            mergedArr[k++] = arr2[j++];
        }

        return mergedArr;
    }
}
```

Output: Merged array: [1, 2, 3, 4, 5, 6, 7, 8]

41. Write a Java program to find the GCD (Greatest Common Divisor) of two numbers.

```
public class GCD {
    public static void main(String[] args) {
        int num1 = 56;
        int num2 = 98;
        System.out.println("GCD of " + num1 + " and " + num2 +
" is: " + findGCD(num1, num2));
    }

    public static int findGCD(int num1, int num2) {
        if (num2 == 0) {
            return num1;
        }
        return findGCD(num2, num1 % num2);
    }
}
```

Output: GCD of 56 and 98 are: 14

42. Write a Java program to find the LCM (Least Common Multiple) of two numbers.

```
public class LCM {
    public static void main(String[] args) {
        int num1 = 12;
        int num2 = 18;
        System.out.println("LCM of " + num1 + " and " + num2 +
" is: " + findLCM(num1, num2));
    }

    public static int findLCM(int num1, int num2) {
        return (num1 * num2) / findGCD(num1, num2);
    }

    public static int findGCD(int num1, int num2) {
        if (num2 == 0) {
            return num1;
        }
        return findGCD(num2, num1 % num2);
    }
}
```

Output: LCM of 12 and 18 is: 36

43. Write a Java program to find the first non-repeating character in a string.

```
import java.util.LinkedHashMap;
import java.util.Map;

public class FirstNonRepeatingCharacter {
    public static void main(String[] args) {
        String str = "swiss";
        System.out.println("First non-repeating character: " +
findFirstNonRepeating(str));
    }
    public static Character findFirstNonRepeating(String str) {
        Map<Character, Integer> charCountMap = new
LinkedHashMap<>();

        for (char c: str.toCharArray()) {
            charCountMap.put(c, charCountMap.getOrDefault(c, 0)
+ 1);
        }
        for (char c: str.toCharArray()) {
            if (charCountMap.get(c) == 1) {
                return c;
            }
        }
        return null;
    }
}
```

Output: First non-repeating character: w

44. Write a Java program to rotate an array to the right by a given number of steps.

```
import java.util.Arrays;

public class RotateArray {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int k = 2;
        rotate(arr, k);
        System.out.println("Array after rotation: " +
Arrays.toString(arr));
    }

    public static void rotate(int[] arr, int k) {
        k = k % arr.length;
        reverse(arr, 0, arr.length - 1);
        reverse(arr, 0, k - 1);
        reverse(arr, k, arr.length - 1);
    }

    public static void reverse(int[] arr, int start, int end) {
        while (start < end) {
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }
}
```

Output: Array after rotation: [4, 5, 1, 2, 3]

45. Write a Java program to reverse the words in a sentence.

```
public class ReverseWords {
    public static void main(String[] args) {
        String sentence = "Hello World from Java";
        String reversed = reverseWords(sentence);
        System.out.println("Reversed sentence: " + reversed);
    }

    public static String reverseWords(String sentence) {
        String[] words = sentence.split(" ");
        StringBuilder reversed = new StringBuilder();

        for (int i = words.length - 1; i >= 0; i--) {
            reversed.append(words[i]).append(" ");
        }

        return reversed.toString().trim();
    }
}
```

Output: Reversed sentence: Java from World Hello

46. Write a Java program using recursion to find the nth Fibonacci number.

```
public class Fibonacci {
    public static void main(String[] args) {
        int n = 10;
        System.out.println("The " + n + "th Fibonacci number
is: " + fib(n));
    }

    public static int fib(int n) {
        if (n <= 1) {
            return n;
        }
        return fib(n - 1) + fib(n - 2);
    }
}
```

Output: The 10th Fibonacci number is: 55