61. Perform Intersection Between Two Arrays

```java
import 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

```
import 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

```
import 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

```
public 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

```
import 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

```
public 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

```
import 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

```
import 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(updatedRecor
ds));
    }
}
```

69. Kaprekar Numbers

```
public 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

```
import 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

```
import 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

```
public 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

```java
import 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

```
public 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

```
public 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

```
import 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

```
import 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

```
public 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

```
public 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

```
public 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

```
public 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

```
public 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

```java
public 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

```
public 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

```
public 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

```
public 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

```
import 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

```
public 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

```
import 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

```
public 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

```
public 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

```
public 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?

```
public 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

```
public 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

```
import 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

```
public 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

```
public 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

```
import 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

```
import 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

```java
public 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

```
public 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

```java
import 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

```
public 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

```
public 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

```
public 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

```
public 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

```
public 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

```
import 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

```
import 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(calcul
ateScores("A")));
        System.out.println(java.util.Arrays.toString(calcul
ateScores("ABC")));
        System.out.println(java.util.Arrays.toString(calcul
ateScores("ABCBACC")));
    }
}
```

```
/*
Output:
[1, 0, 0]
[1, 1, 1]
[2, 2, 3]
*/
```

110. Check if a Number is a Strong Number

```
public 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

```
public 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

```
public 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

```
public 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

```
public 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?

```
import 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

```
public 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

```
public 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

```
public 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

```
import 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

```
import 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

```
public 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

```
public 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

```
import 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

```
import 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

```
import 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

```
public 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

```
import 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

```
public 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

```
public 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

```
public 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

```
public 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"));
    }
}
```