Reverse a String

```
public class StringReversal {
    public static void main(String[] args) {
        String original = "automation";

        // Using StringBuilder
        String reversed = new
StringBuilder(original).reverse().toString();
        System.out.println("Reversed: " + reversed);
    }
}
```

This solution leverages StringBuilder's built-in reverse() method, which is more efficient than manual character manipulation for string reversal operations.

> The output of this code will be:
>
> ```
> Reversed: noitamotua
> ```

by Sreenidhi Rajakrishnan

Palindrome Checker

Problem

Check if the given string reads the same backward as forward.

Technique

Compare original string with its reverse.

Time Complexity

O(n) where n is the string length.

```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "madam";
//if string contains caps and small letters, convert string lower or upper
case and then proceed
        boolean isPalindrome = true;

        for (int i = 0; i < str.length()/2; i++) {
            if (str.charAt(i) != str.charAt(str.length()-i-1)) {
                isPalindrome = false;
                break;
            }
        }
        System.out.println(str + " is palindrome: " + isPalindrome);
    }
}
```

Output:

```
madam is palindrome: true
```

by Sreenidhi Rajakrishnan

Swap Two Numbers

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

        System.out.println("Before: a=" + a + ", b=" + b);

        // Method 1: Using arithmetic operations
        a = a + b;
        b = a - b;
        a = a - b;

        // Method 2: Using XOR (alternative)
        // a = a ^ b;
        // b = a ^ b;
        // a = a ^ b;

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

Output:

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

by Sreenidhi Rajakrishnan

Finding the Largest Number

Initialize

Set first element as the largest number.

Compare

Loop through array and update max if larger number found.

Return

Output the maximum value after full traversal.

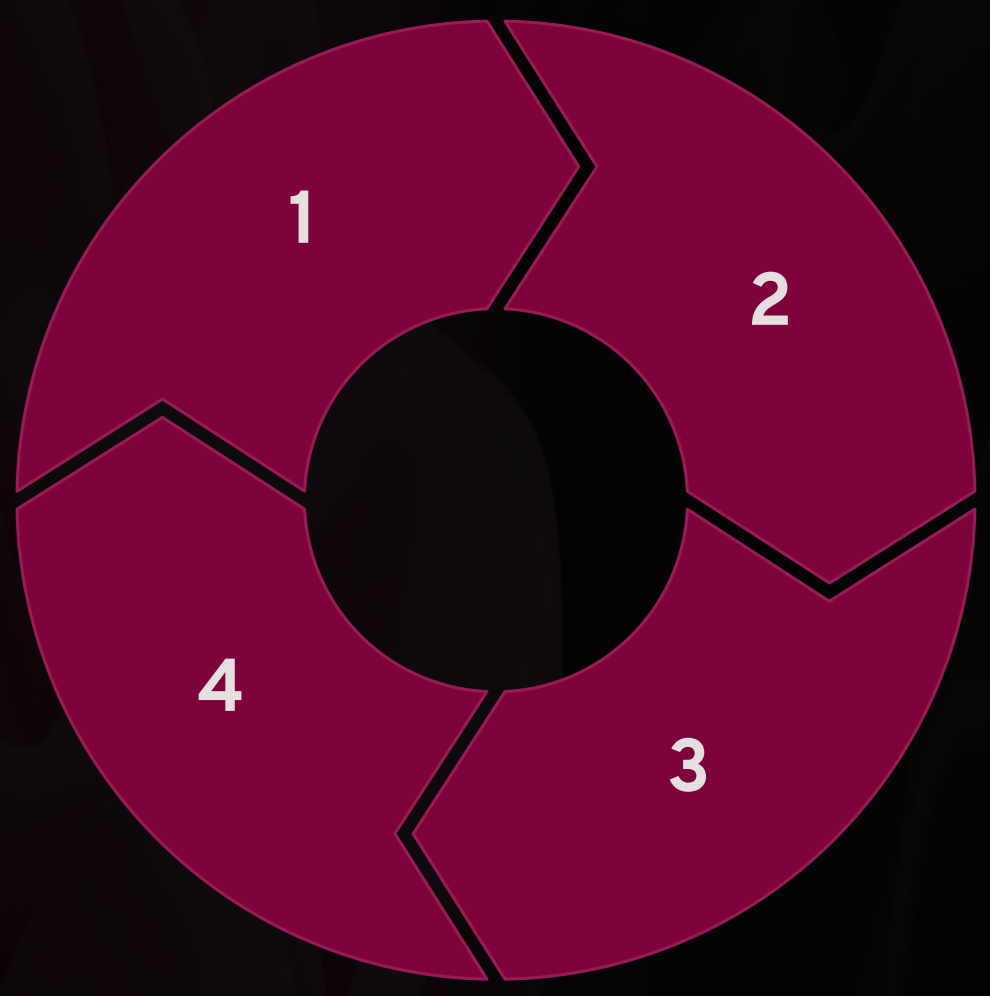
```
public class LargestInArray {
    public static void main(String[] args) {
        int[] numbers = {10, 5, 25, 8, 15, 3};
        int max = numbers[0];

        for (int i = 1; i < numbers.length; i++) {
            if (numbers[i] > max) {
                max = numbers[i];
            }
        }

        System.out.println("Largest number: " + max);
    }
}
```

Output:

```
Largest number: 25
```

by Sreenidhi Rajakrishnan

Finding the Smallest Number



1 Initialize

Set first element as smallest.

2 Compare

Loop through array comparing values.

3 Update

Replace min if smaller found.

4 Output

Return the minimum value.

```
public class SmallestInArray {
    public static void main(String[] args) {
        int[] numbers = {10, 5, 25, 8, 15, 3};
        int min = numbers[0];

        for (int i = 1; i < numbers.length; i++) {
            if (numbers[i] < min) {
                min = numbers[i];
            }
        }

        System.out.println("Smallest number: " + min);
    }
}
```

Output:

```
Smallest number: 3
```

by Sreenidhi Rajakrishnan

Counting Vowels and Consonants

Input

Take a string input to analyze.

Process

Check each character for vowel or consonant.

Count

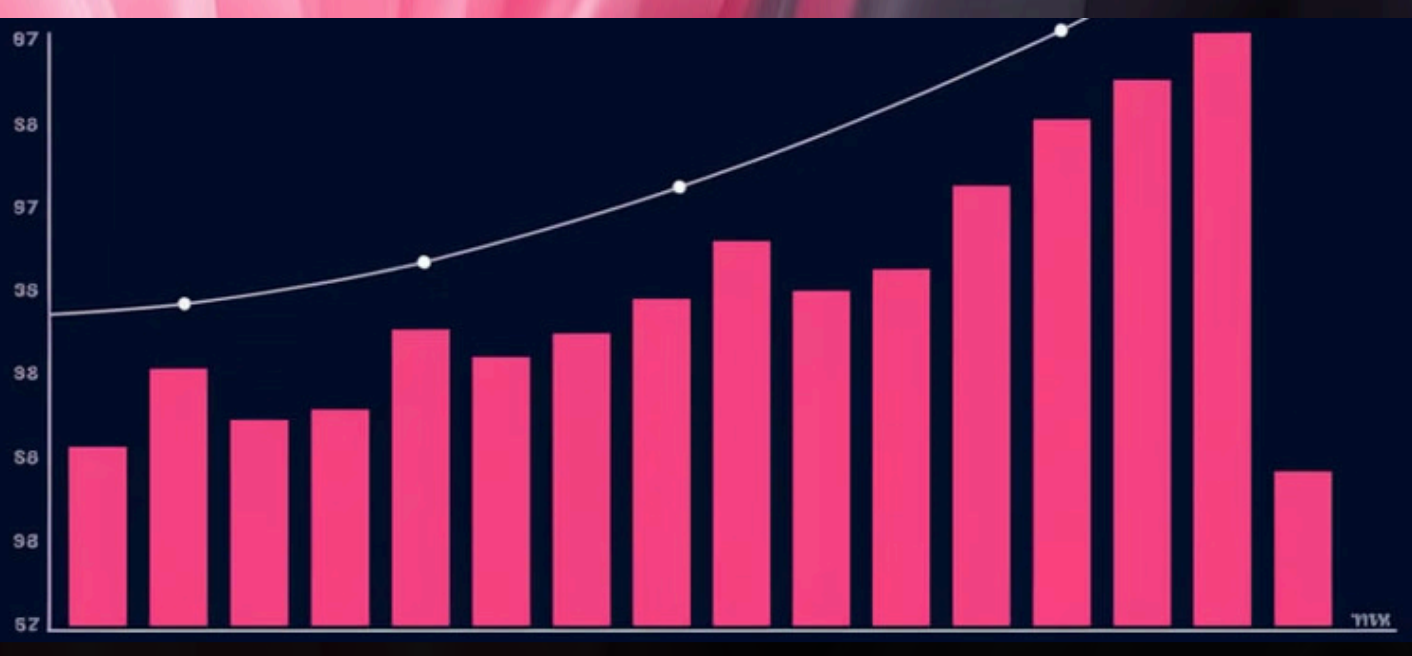
Maintain separate counters for each type.

```
public class VowelConsonantCounter {
    public static void main(String[] args) {
        String str = "Automation World";
        str = str.toLowerCase();
        int vowels = 0, consonants = 0;

        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                    vowels++;
                } else {
                    consonants++;
                }
            }
        }

        System.out.println("Vowels: " + vowels + ", Consonants: " +
consonants);
    }
}
```

Output:

```
Vowels: 6, Consonants: 10
```

by Sreenidhi Rajakrishnan



Character Occurrence Counter

HashMap Approach

Use HashMap to store each character and its count.

Time Complexity

O(n) where n is string length.

Application

Useful for text analysis and data processing.

```
import java.util.HashMap;

public class CharOccurrence {
    public static void main(String[] args) {
        String str = "automation";
        HashMap<Character, Integer> charCount = new HashMap<>();

        for (char ch : str.toCharArray()) {
            if (charCount.containsKey(ch)) {
                charCount.put(ch, charCount.get(ch) + 1);
            } else {
                charCount.put(ch, 1);
            }
        }

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

Output:

```
{a=2, u=2, t=2, o=1, m=1, i=1, n=1}
```

by Sreenidhi Rajakrishnan

Fibonacci Series

1

Initialize

Start with 0 and 1.

2

Calculate

Next number is sum of previous two.

3

Print

Output each Fibonacci number.

4

Continue

Repeat until reaching desired count.

```
public class Fibonacci {
    public static void main(String[] args) {
        int n = 10; // Number of Fibonacci numbers to print
        int a = 0, b = 1;

        System.out.print(a + " " + b + " ");

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

Output:

```
1 1 2 3 5 8 13 21 34 55
```

by Sreenidhi Rajakrishnan

Factorial Calculation

1

Loop Method

Use iteration to multiply numbers.

2

Recursive Method

Function calls itself with decremented value.

3

Base Case

Factorial of 0 or 1 is 1.

```
public class Factorial {
    public static void main(String[] args) {
        int num = 5;

        // Loop method
        int factorial1 = 1;
        for (int i = 1; i <= num; i++) {
            factorial1 *= i;
        }

        // Recursive method
        int factorial2 = factorialRecursive(num);

        System.out.println("Factorial of " + num + ": " + factorial1 + " (loop)");
        System.out.println("Factorial of " + num + ": " + factorial2 + "
(recursive)");
    }

    public static int factorialRecursive(int n) {
        if (n == 0 || n == 1) return 1;
        return n * factorialRecursive(n - 1);
    }
}
```

Output:

```
Factorial of 5: 120 (loop)
Factorial of 5: 120 (recursive)
```

by Sreenidhi Rajakrishnan

Prime Number Check

2

First Prime

Only even prime number.

√n

Check Limit

Only need to check divisors up to square root.

```
public class PrimeChecker {
    public static void main(String[] args) {
        int num = 17;
        boolean isPrime = true;

        if (num <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= Math.sqrt(num); i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        System.out.println(num + " is prime: " + isPrime);
    }
}
```

Output:

```
17 is prime: true
```

by Sreenidhi Rajakrishnan

Sum of Digits

Extract

Get last digit using modulo.

Remove

Remove last digit by dividing.

1 2 3 4

Add

Add digit to running sum.

Repeat

Continue until no digits remain.

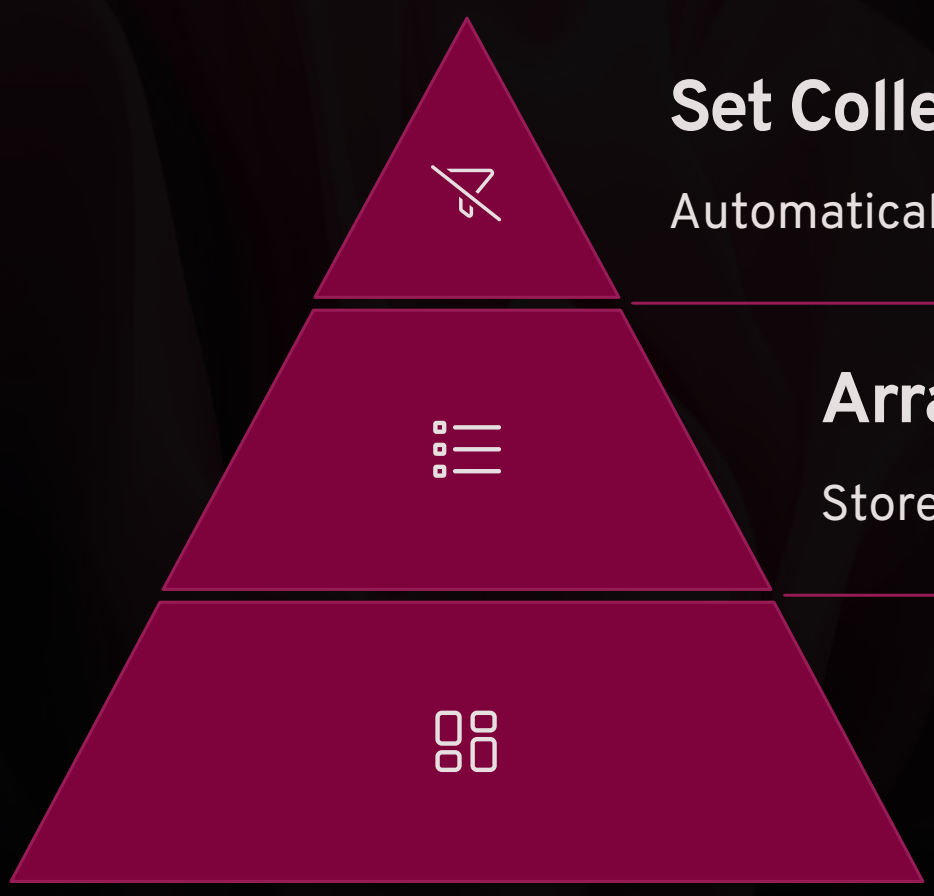
```
public class SumOfDigits {
    public static void main(String[] args) {
        int number = 12345;
        int sum = 0;

        while (number > 0) {
            sum += number % 10;  // Add last digit to sum
            number /= 10;        // Remove last digit
        }

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

Output:

```
Sum of digits: 15
```

by Sreenidhi Rajakrishnan

Remove Duplicates from Array



Set Collection

Automatically removes duplicates

ArrayList

Stores unique elements

Original Array

May contain duplicates

```
import java.util.HashSet;
import java.util.Set;
import java.util.Arrays;

public class RemoveDuplicates {
    public static void main(String[] args) {
        Integer[] numbers = {1, 2, 3, 2, 5, 1, 6, 3, 7};

        // Using HashSet
        Set<Integer> uniqueSet = new HashSet<>(Arrays.asList(numbers));
        Integer[] uniqueNumbers = uniqueSet.toArray(new Integer[0]);

        System.out.println("Original: " + Arrays.toString(numbers));
        System.out.println("Without duplicates: " +
Arrays.toString(uniqueNumbers));
    }
}
```

Output:

```
Original: [1, 2, 3, 2, 5, 1, 6, 3, 7]
Without duplicates: [1, 2, 3, 5, 6, 7]
```

by Sreenidhi Rajakrishnan

Reverse Words in String

```
public class ReverseWords {
    public static void main(String[] args) {
        String sentence = "Java Coding Interview";
        String[] words = sentence.split(" ");
        StringBuilder result = new StringBuilder();

        for (String word : words) {
            StringBuilder reversedWord = new
StringBuilder(word).reverse();
            result.append(reversedWord).append(" ");
        }

        System.out.println("Original: " + sentence);
        System.out.println("Reversed words: " + result.toString().trim());
    }
}
```

Output:

```
Original: Java Coding Interview
Reversed words: avaJ gnidoC weivretnI
```

by Sreenidhi Rajakrishnan

Find Even and Odd Numbers

Input Array	[1, 2, 3, 4, 5, 6, 7, 8, 9]
Even Numbers	[2, 4, 6, 8]
Odd Numbers	[1, 3, 5, 7, 9]
Check Method	num % 2 == 0 (even)

```
import java.util.ArrayList;

public class EvenOddFinder {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};
        ArrayList even = new ArrayList<>();
        ArrayList odd = new ArrayList<>();

        for (int num : numbers) {
            if (num % 2 == 0) {
                even.add(num);
            } else {
                odd.add(num);
            }
        }

        System.out.println("Even numbers: " + even);
        System.out.println("Odd numbers: " + odd);
    }
}
```

Output:

```
Even numbers: [2, 4, 6, 8]
Odd numbers: [1, 3, 5, 7, 9]
```

by Sreenidhi Rajakrishnan

String Length Without .length()

1

Char Array Method

Convert string to character array and count elements.

2

Index Method

Use exception handling to count characters until the end.

3

Manual Count

Iterate through indices until StringIndexOutOfBoundsException.

```
public class StringLengthWithoutMethod {
    public static void main(String[] args) {
        String str = "automation";
        int length = 0;

        try {
            while (true) {
                str.charAt(length);
                length++;
            }
        } catch (StringIndexOutOfBoundsException e) {
            // End of string reached
        }

        System.out.println("Length of string: " + length);
    }
}
```

Output:

```
Length of string: 10
```

by Sreenidhi Rajakrishnan

String to Integer Conversion

String to Integer

```
String str = "123";
int num = Integer.parseInt(str);
// Or
int num2 = Integer.valueOf(str);
```

Integer to String

```
int number = 123;
String str1 =
Integer.toString(number);
// Or
String str2 =
String.valueOf(number);
// Or
String str3 = "" + number;
```

```
public class StringIntConversion {
    public static void main(String[] args) {
        // String to Integer
        String numStr = "12345";
        int num = Integer.parseInt(numStr);
        System.out.println("String to Integer: " + num);

        // Integer to String
        int number = 12345;
        String str = Integer.toString(number);
        System.out.println("Integer to String: " + str);
    }
}
```

Output:

```
String to Integer: 12345
Integer to String: 12345
```

by Sreenidhi Rajakrishnan

Print Elements at Even/Odd Indexes

1

Understand Indexes

Array indexes start at 0 (even).

2

Set Up Collections

Create separate lists for even and odd indexed elements.

3

Iterate Array

Check index modulo 2 to determine even or odd.

4

Display Results

Print both collections of elements.

```
public class EvenOddIndexElements {
    public static void main(String[] args) {
        String[] elements = {"Java", "Selenium", "TestNG", "Maven", "Jenkins",
"Docker"};

        System.out.print("Even index elements: ");
        for (int i = 0; i < elements.length; i += 2) {
            System.out.print(elements[i] + " ");
        }

        System.out.print("\nOdd index elements: ");
        for (int i = 1; i < elements.length; i += 2) {
            System.out.print(elements[i] + " ");
        }
    }
}
```

Output:

```
Even index elements: Java TestNG Jenkins
Odd index elements: Selenium Maven Docker
```

by Sreenidhi Rajakrishnan

Array Reversal

Reverse Elements

To reverse the array, create a new array and iterate from the end of the original array to the beginning, assigning elements to the new array in reverse order.

Display Reversed Array

Print the reversed array after iterating through and reversing the elements.

```
import java.util.Arrays;

public class ArrayReversal {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};

        System.out.println("Original: " + Arrays.toString(array));

        int start = 0;
        int end = array.length - 1;

        while (start < end) {
            // Swap elements
            int temp = array[start];
            array[start] = array[end];
            array[end] = temp;

            // Move pointers
            start++;
            end--;
        }

        System.out.println("Reversed: " + Arrays.toString(array));
    }
}
```

Output:

```
Original: [1, 2, 3, 4, 5]
Reversed: [5, 4, 3, 2, 1]
```

by Sreenidhi Rajakrishnan

Check if Array is Sorted

To check if an array is sorted, loop through the array and compare each element with the next one in sequence. If any element is greater than the next, return false indicating the array is not sorted. Otherwise, return true if the loop completes without finding any unsorted pairs.

```
public class SortedArrayCheck {
    public static void main(String[] args) {
        int[] array = {1, 2, 4, 7, 9}; // Sorted array
        //int[] array = {1, 5, 3, 7, 9}; // Unsorted array
        boolean isSorted = true;

        for (int i = 0; i < array.length - 1; i++) {
            if (array[i] > array[i + 1]) {
                isSorted = false;
                break;
            }
        }

        System.out.println("Array is sorted: " + isSorted);
    }
}
```

Output:

```
Array is sorted: true
```

by Sreenidhi Rajakrishnan

Case Conversion

Using String Methods

```
String str = "Hello";
String upper =
str.toUpperCase();
String lower = str.toLowerCase();
```

Manual Character Conversion

```
// ASCII value difference: 32
// 'A' (65) to 'a' (97)
char upper = 'a';
char lower = (char)(upper - 32);
```

```
public class CaseConversion {
    public static void main(String[] args) {
        String input = "Java Programming";
        StringBuilder result = new StringBuilder();

        for (char c : input.toCharArray()) {
            if (Character.isUpperCase(c)) {
                result.append(Character.toLowerCase(c));
            } else if (Character.isLowerCase(c)) {
                result.append(Character.toUpperCase(c));
            } else {
                result.append(c);
            }
        }

        System.out.println("Original: " + input);
        System.out.println("Case converted: " + result.toString());
    }
}
```

Output:

```
Original: Java Programming
Case converted: jAVA pROGRAMMING
```

by Sreenidhi Rajakrishnan