

Find Second Largest Number in an Array



Sort array

One approach is to sort and find second-to-last element

Track two variables

Better approach: single-pass with two variables

Implement solution

Use a loop that tracks largest and second largest

```
public class SecondLargestFinder {  
    public static int findSecondLargest(int[] arr) {  
        int largest = Integer.MIN_VALUE;  
        int secondLargest = Integer.MIN_VALUE;  
  
        for (int num : arr) {  
            if (num > largest) {  
                secondLargest = largest;  
                largest = num;  
            } else if (num > secondLargest && num != largest) {  
                secondLargest = num;  
            }  
        }  
        return secondLargest;  
    }  
  
    public static void main(String[] args) {  
        int[] numbers = {12, 35, 1, 10, 34, 1};  
        System.out.println("Second largest number: " +  
findSecondLargest(numbers));  
    }  
}
```

Output

Second largest number: 34



by Sreenidhi Rajakrishnan

Find Duplicate Elements in an Array



Hash Set

Use HashSet to track seen elements



Single Pass

Process array elements just once



Store Duplicates

Add duplicates to result list

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class DuplicateFinder {
    public static List<Integer> findDuplicates(int[] arr) {
        Set<Integer> seen = new HashSet<>();
        List<Integer> duplicates = new ArrayList<>();

        for (int num : arr) {
            if (!seen.add(num)) { // add() returns false if element already exists
                duplicates.add(num);
            }
        }
        return duplicates;
    }

    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 2, 7, 8, 8, 3};
        System.out.println("Duplicate elements: " + findDuplicates(numbers));
    }
}
```

Output

Duplicate elements: [2, 3, 8]



by Sreenidhi Rajakrishnan

Check if Two Strings are Anagrams

What are anagrams?

Two strings containing same characters with same frequency but different order.

Examples: "listen" and "silent",
"triangle" and "integral"

Solution approaches

1. Sort both strings and compare
2. Count character frequencies
3. Use character array for counting

```
public class AnagramChecker {  
    public static boolean areAnagrams(String str1, String str2) {  
        // Remove spaces and convert to lowercase  
        str1 = str1.replaceAll("\\s", "").toLowerCase();  
        str2 = str2.replaceAll("\\s", "").toLowerCase();  
  
        // Check if lengths are different  
        if (str1.length() != str2.length()) {  
            return false;  
        }  
  
        // Convert to char arrays and sort  
        char[] charArray1 = str1.toCharArray();  
        char[] charArray2 = str2.toCharArray();  
        java.util.Arrays.sort(charArray1);  
        java.util.Arrays.sort(charArray2);  
  
        // Compare sorted arrays  
        return java.util.Arrays.equals(charArray1, charArray2);  
    }  
  
    public static void main(String[] args) {  
        String s1 = "Listen";  
        String s2 = "Silent";  
        System.out.println("'" + s1 + "' and '" + s2 + "' are anagrams: "  
            + areAnagrams(s1, s2));  
    }  
}
```

Output

"Listen" and "Silent" are anagrams: true



by Sreenidhi Rajakrishnan

Find the Missing Number in a Sequence

1

Start

Begin with sequence 1 to N

N

End

Expected to have all numbers

N-1

Given

Array contains all except one number

?

Find

Identify the missing number

```
public class MissingNumberFinder {  
    public static int findMissingNumber(int[] arr, int n) {  
        // Expected sum of numbers from 1 to n  
        int expectedSum = n * (n + 1) / 2;  
  
        // Calculate actual sum of array  
        int actualSum = 0;  
        for (int num : arr) {  
            actualSum += num;  
        }  
  
        // Missing number is the difference  
        return expectedSum - actualSum;  
    }  
  
    public static void main(String[] args) {  
        int[] numbers = {1, 2, 4, 6, 3, 7, 8};  
        int n = 8; // Range is 1 to 8  
        System.out.println("Missing number: " +  
            findMissingNumber(numbers, n));  
    }  
}
```

Output

Missing number: 5



by Sreenidhi Rajakrishnan

Sort an Array Without Using `Arrays.sort()`

Choose a Sorting Algorithm

We'll implement Bubble Sort as a simple example.

Implement the Algorithm

Create nested loops to compare and swap elements.

Test with Sample Data

Verify the array is correctly sorted.

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

Output

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



by Sreenidhi Rajakrishnan

Find the First Non-Repeated Character in a String

Character Frequency Map

Use HashMap to track character counts in the string.

Two-Pass Approach

First count occurrences. Then find first character with count of 1.

Time Complexity

O(n) where n is the length of the string.

```
import java.util.HashMap;
import java.util.Map;

public class FirstNonRepeatedChar {
    public static char findFirstNonRepeatedChar(String str) {
        Map<Character, Integer> charCounts = new HashMap<>();

        // Count occurrences of each character
        for (char c : str.toCharArray()) {
            charCounts.put(c, charCounts.getOrDefault(c, 0) + 1);
        }

        // Find first character with count 1
        for (char c : str.toCharArray()) {
            if (charCounts.get(c) == 1) {
                return c;
            }
        }

        // If no non-repeated character found
        return '\0';
    }

    public static void main(String[] args) {
        String str = "programming";
        char result = findFirstNonRepeatedChar(str);

        if (result != '\0') {
            System.out.println("First non-repeated character: " + result);
        } else {
            System.out.println("No non-repeated character found");
        }
    }
}
```

Output

First non-repeated character: p



by Sreenidhi Rajakrishnan

Find Common Elements in Two Arrays

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

public class CommonElementsFinder {
    public static Integer[] findCommonElements(Integer[] arr1, Integer[]
arr2) {
        Set<Integer> set1 = new HashSet<>(Arrays.asList(arr1));
        Set<Integer> commonElements = new HashSet<>();

        for (Integer num : arr2) {
            if (set1.contains(num)) {
                commonElements.add(num);
            }
        }

        return commonElements.toArray(new Integer[0]);
    }

    public static void main(String[] args) {
        Integer[] array1 = {1, 2, 3, 4, 5};
        Integer[] array2 = {3, 4, 5, 6, 7};

        Integer[] common = findCommonElements(array1, array2);
        System.out.println("Common elements: " + Arrays.toString(common));
    }
}
```

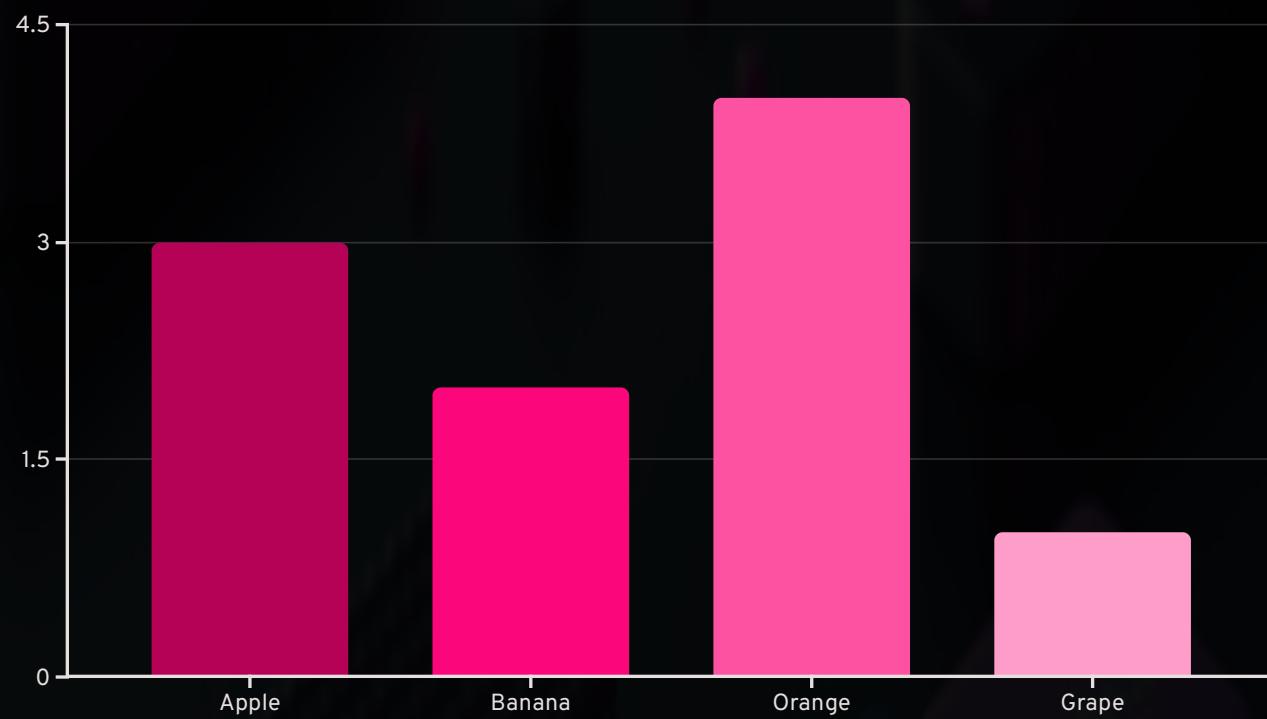
Output

Common elements: [3, 4, 5]



by Sreenidhi Rajakrishnan

Count Frequency of Elements Using HashMap



```
import java.util.HashMap;
import java.util.Map;

public class FrequencyCounter {
    public static Map<String, Integer> countFrequency(String[] array) {
        Map<String, Integer> frequencyMap = new HashMap<>();

        for (String item : array) {
            // If key exists, increment count; otherwise, set count to 1
            frequencyMap.put(item, frequencyMap.getOrDefault(item, 0) + 1);
        }

        return frequencyMap;
    }

    public static void main(String[] args) {
        String[] fruits = {"Apple", "Banana", "Apple", "Orange", "Banana",
                           "Orange", "Orange", "Apple", "Orange", "Grape"};

        Map<String, Integer> frequency = countFrequency(fruits);

        System.out.println("Frequency of elements:");
        for (Map.Entry<String, Integer> entry : frequency.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }
}
```

Output

```
Frequency of elements:
Apple: 3
Grape: 1
Orange: 4
Banana: 2
```



by Sreenidhi Rajakrishnan

Reverse Words in a Sentence



Split sentence into words

Use string split method



Reverse word order

Iterate from end to start



Join words back together

Use space as delimiter

```
public class WordReverser {  
    public static String reverseWords(String sentence) {  
        // Split the sentence into words  
        String[] words = sentence.split("\\s+");  
        StringBuilder reversed = new StringBuilder();  
  
        // Add words in reverse order  
        for (int i = words.length - 1; i >= 0; i--) {  
            reversed.append(words[i]);  
            if (i > 0) {  
                reversed.append(" ");  
            }  
        }  
  
        return reversed.toString();  
    }  
  
    public static void main(String[] args) {  
        String sentence = "Java is a programming language";  
        System.out.println("Original: " + sentence);  
        System.out.println("Reversed: " + reverseWords(sentence));  
    }  
}
```

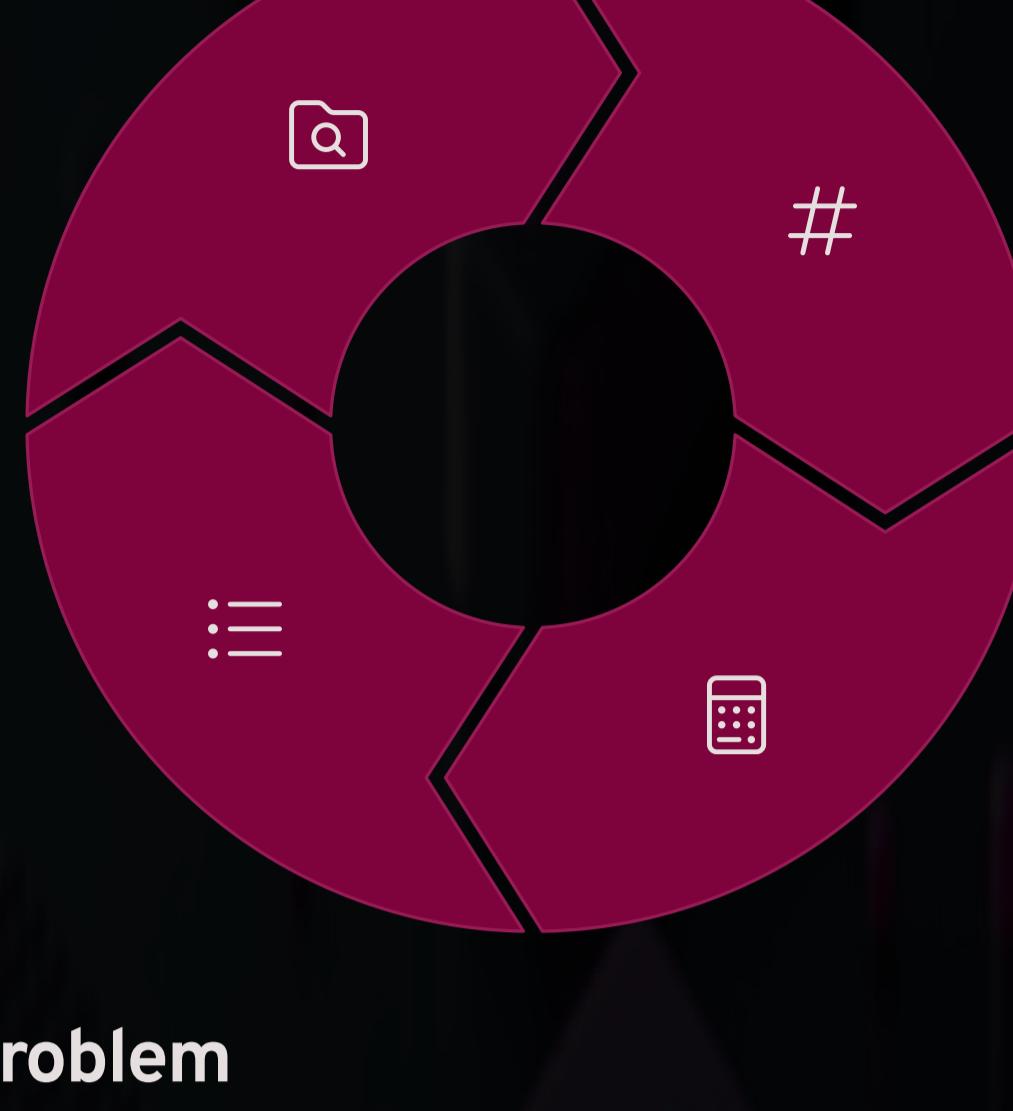
Output

Original: Java is a programming language
Reversed: language programming a is Java



by Sreenidhi Rajakrishnan

Find Pairs in Array Whose Sum Equals a Target



Define Problem

Find all pairs (a,b) where $a+b$ equals target sum



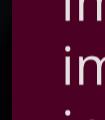
Use Hash Set

Store visited numbers for $O(n)$ lookup



Check Complement

For each number, look for (target-number)



Collect Pairs

Add matching pairs to result list

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class PairSumFinder {
    public static List<int[]> findPairsWithSum(int[] arr, int targetSum) {
        List<int[]> pairs = new ArrayList<>();
        Set<Integer> visitedNumbers = new HashSet<>();

        for (int num : arr) {
            int complement = targetSum - num;

            if (visitedNumbers.contains(complement)) {
                pairs.add(new int[]{complement, num});
            }

            visitedNumbers.add(num);
        }

        return pairs;
    }

    public static void main(String[] args) {
        int[] numbers = {2, 4, 3, 5, 6, -2, 8, 7, 1};
        int target = 6;

        List<int[]> pairs = findPairsWithSum(numbers, target);

        System.out.println("Pairs with sum " + target + ":");
        for (int[] pair : pairs) {
            System.out.println("(" + pair[0] + ", " + pair[1] + ")");
        }
    }
}
```

Output

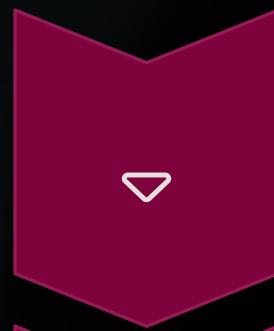
Pairs with sum 6:

(2, 4)
(-2, 8)
(5, 1)



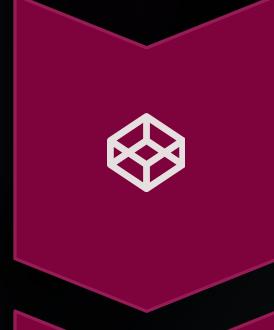
by Sreenidhi Rajakrishnan

Merge Two Sorted Arrays



Start with two sorted arrays

Arrays are already in order



Use merge method

Similar to merge sort's merge step



Create merged array

Result maintains sorted order

```
public class MergeSortedArrays {  
    public static int[] merge(int[] arr1, int[] arr2) {  
        int n1 = arr1.length;  
        int n2 = arr2.length;  
        int[] result = new int[n1 + n2];  
  
        int i = 0, j = 0, k = 0;  
  
        // Compare elements from both arrays and add smaller one to result  
        while (i < n1 && j < n2) {  
            if (arr1[i] <= arr2[j]) {  
                result[k++] = arr1[i++];  
            } else {  
                result[k++] = arr2[j++];  
            }  
        }  
  
        // Copy remaining elements from arr1, if any  
        while (i < n1) {  
            result[k++] = arr1[i++];  
        }  
  
        // Copy remaining elements from arr2, if any  
        while (j < n2) {  
            result[k++] = arr2[j++];  
        }  
  
        return result;  
    }  
  
    public static void main(String[] args) {  
        int[] arr1 = {1, 3, 5, 7};  
        int[] arr2 = {2, 4, 6, 8, 10};  
  
        int[] merged = merge(arr1, arr2);  
  
        System.out.println("Merged array: " +  
            java.util.Arrays.toString(merged));  
    }  
}
```

Output

Merged array: [1, 2, 3, 4, 5, 6, 7, 8, 10]



by Sreenidhi Rajakrishnan

Convert a List to Set and Vice Versa



List to Set Conversion

Pass List to Set constructor to remove duplicates



Set to List Conversion

Pass Set to List constructor to allow duplicates and indexing



Use Cases

Remove duplicates or restore original order



Key Differences

List allows duplicates and maintains order; Set has unique elements

```
import java.util.*;  
  
public class CollectionConverter {  
    public static void main(String[] args) {  
        // Create List with duplicates  
        List<String> namesList = new ArrayList<>();  
        namesList.add("Alice");  
        namesList.add("Bob");  
        namesList.add("Alice"); // Duplicate  
        namesList.add("Charlie");  
  
        System.out.println("Original List: " + namesList);  
  
        // Convert List to Set (removes duplicates)  
        Set<String> namesSet = new HashSet<>(namesList);  
        System.out.println("After List to Set conversion: " + namesSet);  
  
        // Convert Set back to List  
        List<String> uniqueNamesList = new ArrayList<>(namesSet);  
        System.out.println("After Set to List conversion: " + uniqueNamesList);  
    }  
}
```

Output

```
Original List: [Alice, Bob, Alice, Charlie]  
After List to Set conversion: [Bob, Alice, Charlie]  
After Set to List conversion: [Bob, Alice, Charlie]
```



by Sreenidhi Rajakrishnan

Use of ArrayList, HashSet, and HashMap in Code

```
import java.util.*;

public class CollectionsDemo {
    public static void main(String[] args) {
        // ArrayList demo
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Apple"); // Allows duplicates
        System.out.println("ArrayList: " + fruits);

        // HashSet demo
        HashSet<String> uniqueFruits = new HashSet<>();
        uniqueFruits.add("Apple");
        uniqueFruits.add("Banana");
        uniqueFruits.add("Apple"); // Duplicate not added
        System.out.println("HashSet: " + uniqueFruits);

        // HashMap demo
        HashMap<String, Integer> fruitCounts = new HashMap<>();
        fruitCounts.put("Apple", 5);
        fruitCounts.put("Banana", 3);
        fruitCounts.put("Orange", 2);
        System.out.println("HashMap: " + fruitCounts);
        System.out.println("Apple count: " + fruitCounts.get("Apple"));
    }
}
```

Output

```
ArrayList: [Apple, Banana, Apple]
HashSet: [Apple, Banana]
HashMap: {Apple=5, Orange=2, Banana=3}
Apple count: 5
```



by Sreenidhi Rajakrishnan

Remove All White Spaces from a String

Input	Output
"Hello World"	"HelloWorld"
" Java Programming "	"JavaProgramming"
"No Spaces Here"	"NoSpacesHere"

```
public classWhiteSpaceRemover {  
    // Method 1: Using replace() with regular expression  
    public static String removeWhiteSpacesRegex(String input) {  
        return input.replaceAll("\\s+", "");  
    }  
  
    // Method 2: Manual approach with StringBuilder  
    public static String removeWhiteSpacesManual(String input) {  
        StringBuilder result = new StringBuilder();  
  
        for (char c : input.toCharArray()) {  
            if (!Character.isWhitespace(c)) {  
                result.append(c);  
            }  
        }  
  
        return result.toString();  
    }  
  
    public static void main(String[] args) {  
        String text = " Java Programming is fun ";  
  
        System.out.println("Original string: " + text + " ");  
        System.out.println("After removing spaces (regex): " +  
            removeWhiteSpacesRegex(text) + " ");  
        System.out.println("After removing spaces (manual): " +  
            removeWhiteSpacesManual(text) + " ");  
        System.out.println("Using String Replace "+text.replace(" ", ""));  
    }  
}
```

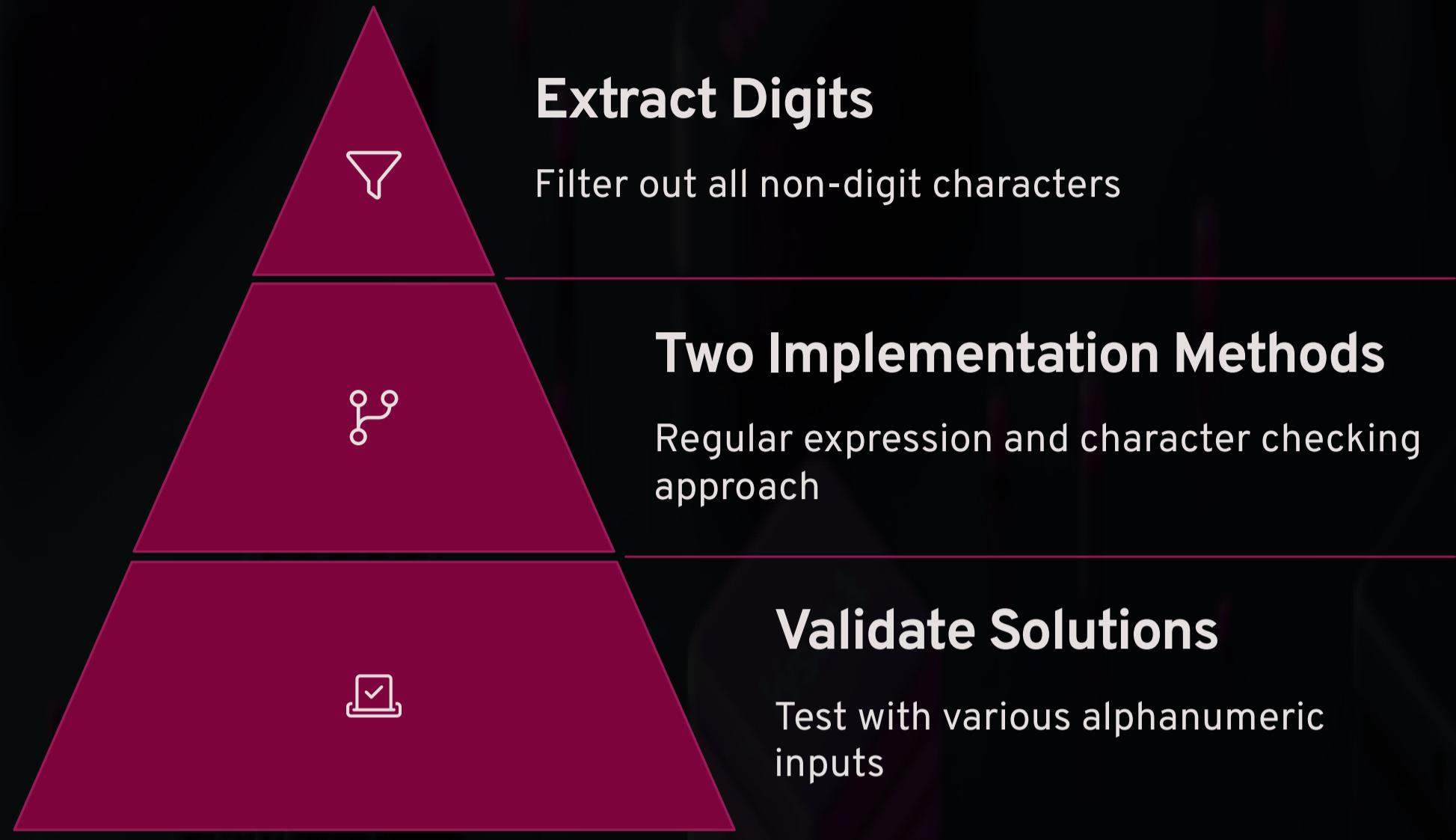
Output

```
Original string: " Java Programming is fun "  
After removing spaces (regex): "JavaProgrammingisfun"  
After removing spaces (manual): "JavaProgrammingisfun"  
Using String Replace: "JavaProgrammingisfun"
```



by Sreenidhi Rajakrishnan

Extract Only Digits from an Alphanumeric String



```
public class DigitExtractor {  
    // Method 1: Using regular expressions  
    public static String extractDigitsRegex(String input) {  
        return input.replaceAll("[^0-9]", "");  
    }  
  
    // Method 2: Using Character.isDigit()  
    public static String extractDigitsManual(String input) {  
        StringBuilder result = new StringBuilder();  
  
        for (char c : input.toCharArray()) {  
            if (Character.isDigit(c)) {  
                result.append(c);  
            }  
        }  
  
        return result.toString();  
    }  
  
    public static void main(String[] args) {  
        String alphanumeric = "abc123def456ghi789";  
  
        System.out.println("Original string: " + alphanumeric);  
        System.out.println("Extracted digits (regex): " +  
            extractDigitsRegex(alphanumeric));  
        System.out.println("Extracted digits (manual): " +  
            extractDigitsManual(alphanumeric));  
    }  
}
```

Output

```
Original string: abc123def456ghi789  
Extracted digits (regex): 123456789  
Extracted digits (manual): 123456789
```



by Sreenidhi Rajakrishnan

Read Data from Excel Using Apache POI

Dependencies

Add Apache POI libraries to your project.

- poi-5.2.3.jar
- poi-ooxml-5.2.3.jar
- commons-io-2.11.0.jar

Core Classes

- XSSFWorkbook: For XLSX files
- HSSFWorkbook: For XLS files
- Sheet, Row, Cell: For data access

```
import java.io.File;
import java.io.FileInputStream;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelReader {
    public static void main(String[] args) {
        try {
            FileInputStream file = new FileInputStream(new File("data.xlsx"));

            // Create Workbook instance for XLSX file
            Workbook workbook = new XSSFWorkbook(file);

            // Get first sheet
            Sheet sheet = workbook.getSheetAt(0);

            // Iterate through rows
            for (Row row : sheet) {
                // Iterate through cells in row
                for (Cell cell : row) {
                    switch (cell.getCellType()) {
                        case STRING:
                            System.out.print(cell.getStringCellValue() + "\t");
                            break;
                        case NUMERIC:
                            System.out.print(cell.getNumericCellValue() + "\t");
                            break;
                        case BOOLEAN:
                            System.out.print(cell.getBooleanCellValue() + "\t");
                            break;
                        default:
                            System.out.print("\t");
                    }
                }
                System.out.println();
            }

            file.close();
            workbook.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output

Name	Age	Department
John	30.0	Engineering
Alice	25.0	Marketing
Bob	35.0	Finance



by Sreenidhi Rajakrishnan

Capture a Screenshot in Selenium Using Java



Set up WebDriver instance

Initialize and configure Chrome/Firefox driver



Navigate to target webpage

Load the page you want to capture



Take screenshot using TakesScreenshot

Cast WebDriver to TakesScreenshot interface



Save captured image to disk

Use FileUtils to write image to file

```
import java.io.File;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class ScreenshotCapture {
    public static void main(String[] args) {
        // Set path to ChromeDriver
        System.setProperty("webdriver.chrome.driver",
        "path/to/chromedriver");

        // Initialize WebDriver
        WebDriver driver = new ChromeDriver();

        try {
            // Navigate to website
            driver.get("https://www.example.com");

            // Take screenshot
            TakesScreenshot scrShot = (TakesScreenshot) driver;
            File srcFile = scrShot.getScreenshotAs(OutputType.FILE);

            // Save screenshot
            File destFile = new File("screenshot.png");
            FileUtils.copyFile(srcFile, destFile);

            System.out.println("Screenshot captured and saved to: " +
                destFile.getAbsolutePath());

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // Close browser
            driver.quit();
        }
    }
}
```

Output

Screenshot captured and saved to: C:\path\to\project\screenshot.png



by Sreenidhi Rajakrishnan

Implement Implicit, Explicit, and Fluent Waits



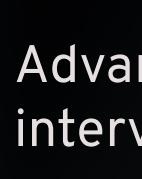
Implicit Wait

Global timeout for all elements. Polls DOM until element found or timeout.



Explicit Wait

Waits for specific condition. More precise than implicit wait.



Fluent Wait

Advanced wait with custom polling interval and exception ignoring.

```
import org.openqa.selenium.By;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.FluentWait;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.function.Function;

public class SeleniumWaits {
    public static void main(String[] args) {
        // Set path to ChromeDriver
        System.setProperty("webdriver.chrome.driver",
        "path/to/chromedriver");

        // Initialize WebDriver
        WebDriver driver = new ChromeDriver();

        try {
            // 1. Implicit Wait
            driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

            driver.get("https://www.example.com");
            // Element will be searched for up to 10 seconds
            WebElement implicitElement = driver.findElement(By.id("someId"));

            // 2. Explicit Wait
            WebDriverWait explicitWait = new WebDriverWait(driver,
            Duration.ofSeconds(20));
            WebElement explicitElement = explicitWait.until(
                ExpectedConditions.visibilityOfElementLocated(By.id("loadingElement"))
            );

            // 3. Fluent Wait
            Wait<WebDriver> fluentWait = new FluentWait<WebDriver>(driver)
                .withTimeout(Duration.ofSeconds(30))
                .pollingEvery(Duration.ofMillis(500))
                .ignoring(NoSuchElementException.class);

            WebElement fluentElement = fluentWait.until(new
            Function<WebDriver, WebElement>() {
                public WebElement apply(WebDriver driver) {
                    return driver.findElement(By.id("dynamicElement"));
                }
            });
        }

        System.out.println("All wait examples executed successfully");

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        driver.quit();
    }
}
```

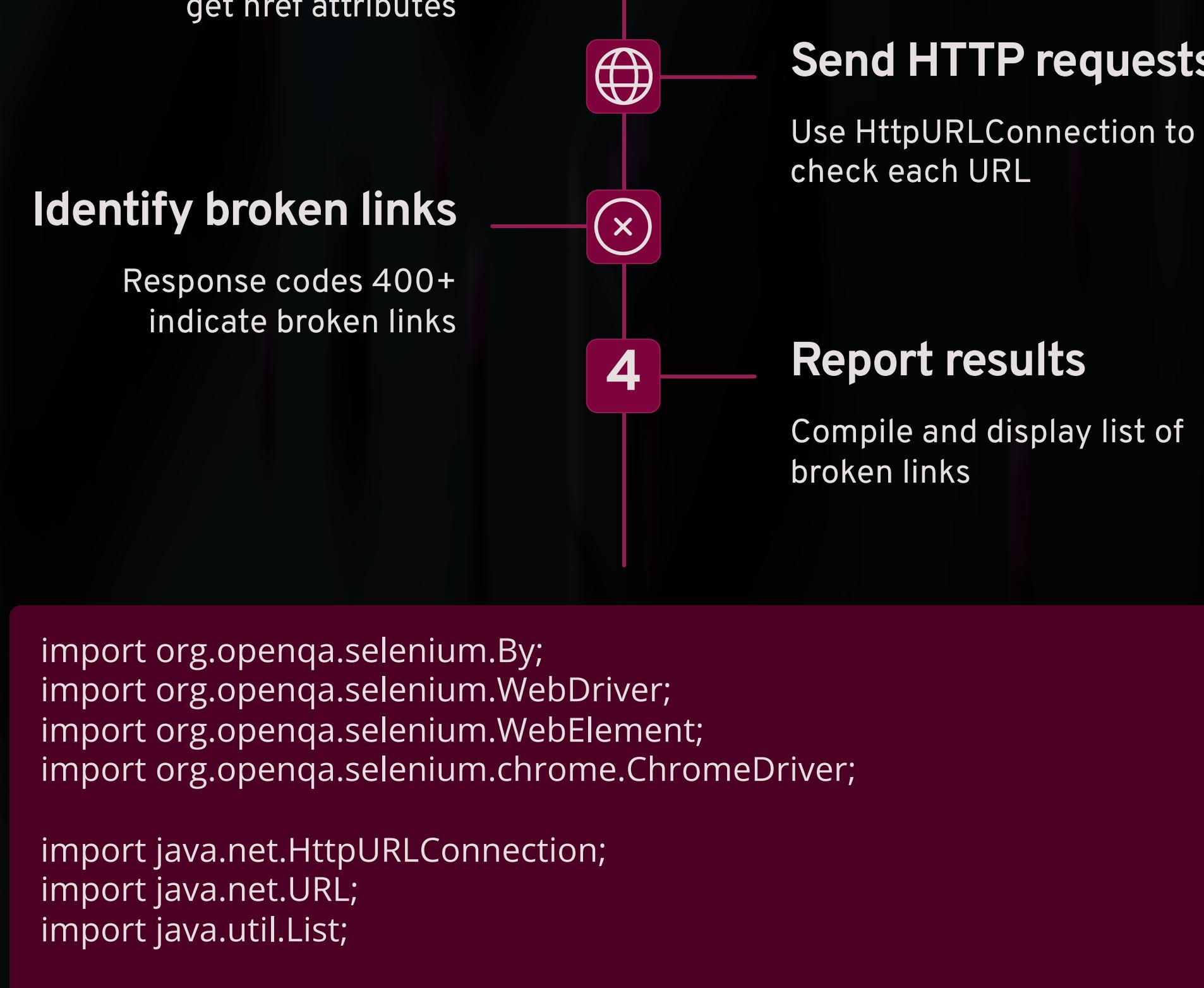
Output

All wait examples executed successfully



by Sreenidhi Rajakrishnan

Find All Broken Links on a Webpage



```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import java.net.HttpURLConnection;
import java.net.URL;
import java.util.List;

public class BrokenLinkFinder {
    public static void main(String[] args) {
        // Set path to ChromeDriver
        System.setProperty("webdriver.chrome.driver",
        "path/to/chromedriver");

        // Initialize WebDriver
        WebDriver driver = new ChromeDriver();

        try {
            // Navigate to website
            driver.get("https://www.example.com");

            // Find all links on the page
            List<WebElement> links = driver.findElements(By.tagName("a"));
            System.out.println("Total links found: " + links.size());

            int brokenLinks = 0;

            // Check each link
            for (WebElement link : links) {
                String url = link.getAttribute("href");

                if (url == null || url.isEmpty()) {
                    System.out.println("URL is empty or null");
                    continue;
                }

                // Skip non-HTTP URLs
                if (!url.startsWith("http")) {
                    System.out.println("URL is not HTTP: " + url);
                    continue;
                }

                try {
                    // Create connection
                    HttpURLConnection connection = (HttpURLConnection) new
URL(url).openConnection();
                    connection.setRequestMethod("HEAD");
                    connection.connect();

                    int responseCode = connection.getResponseCode();

                    if (responseCode >= 400) {
                        System.out.println("Broken link: " + url + " - Response code: "
+ responseCode);
                        brokenLinks++;
                    } else {
                        System.out.println("Valid link: " + url);
                    }
                } catch (Exception e) {
                    System.out.println("Exception checking link: " + url + " - " +
e.getMessage());
                    brokenLinks++;
                }
            }

            System.out.println("Total broken links found: " + brokenLinks);

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            driver.quit();
        }
    }
}
```

Output

```
Total links found: 14
Valid link: https://www.example.com/about
Valid link: https://www.example.com/contact
Broken link: https://www.example.com/old-page - Response code: 404
Valid link: https://www.example.com/products
Valid link: https://www.example.com/services
...
Total broken links found: 3
```



by Sreenidhi Rajakrishnan

Count Web Elements on a Webpage

Selenium WebDriver can easily count various elements on a webpage. The code below finds and counts all links, images, and buttons.

Initialize WebDriver

Create a Chrome WebDriver instance to control the browser.

Navigate to Website

Open the target webpage using the `get()` method.

Find Elements

Use `findElements()` with appropriate tag selectors to locate elements.

Count and Report

Get the `size()` of each collection and display the results.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class CountWebElements {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        try {
            driver.get("https://www.example.com");

            // Count different elements
            int linkCount = driver.findElements(By.tagName("a")).size();
            int imageCount = driver.findElements(By.tagName("img")).size();
            int buttonCount = driver.findElements(By.tagName("button")).size();

            // Print results
            System.out.println("Number of links: " + linkCount);
            System.out.println("Number of images: " + imageCount);
            System.out.println("Number of buttons: " + buttonCount);
            System.out.println("Total elements counted: " + (linkCount +
imageCount + buttonCount));

        } finally {
            driver.quit();
        }
    }
}
```

Output

```
Number of links: 24
Number of images: 15
Number of buttons: 8
Total elements counted: 47
```



by Sreenidhi Rajakrishnan