Day-6(Java 8+ features ):

Java-8-17 Overview

-----------------------------------------------------------------------------------------------------------------------------------------------

Q1.Sum of a List of Integers

Emma is a software developer working on a financial application. She needs to calculate the total sum of several transactions entered by

the user. Help Emma write a program that reads a list of integers representing transaction amounts and calculates their total sum using

Java Streams.

Test Case 1:

Input: 100 200 300 400 500

Output: Total Sum: 1500

Test Case 2:

Input: 10 20 30

Output: Total Sum: 60

Solution:

import java.util.\*;

import java.util.stream.\*;

public class SumOfIntegers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a list of integers separated by space:");

List<Integer> numbers = Arrays.stream(scanner.nextLine().split("\\s+"))

.map(Integer::parseInt)

.collect(Collectors.toList());

// Calculate total sum

int totalSum = calculateSum(numbers);

System.out.println("Total Sum: " + totalSum);

}

public static int calculateSum(List<Integer> numbers) {

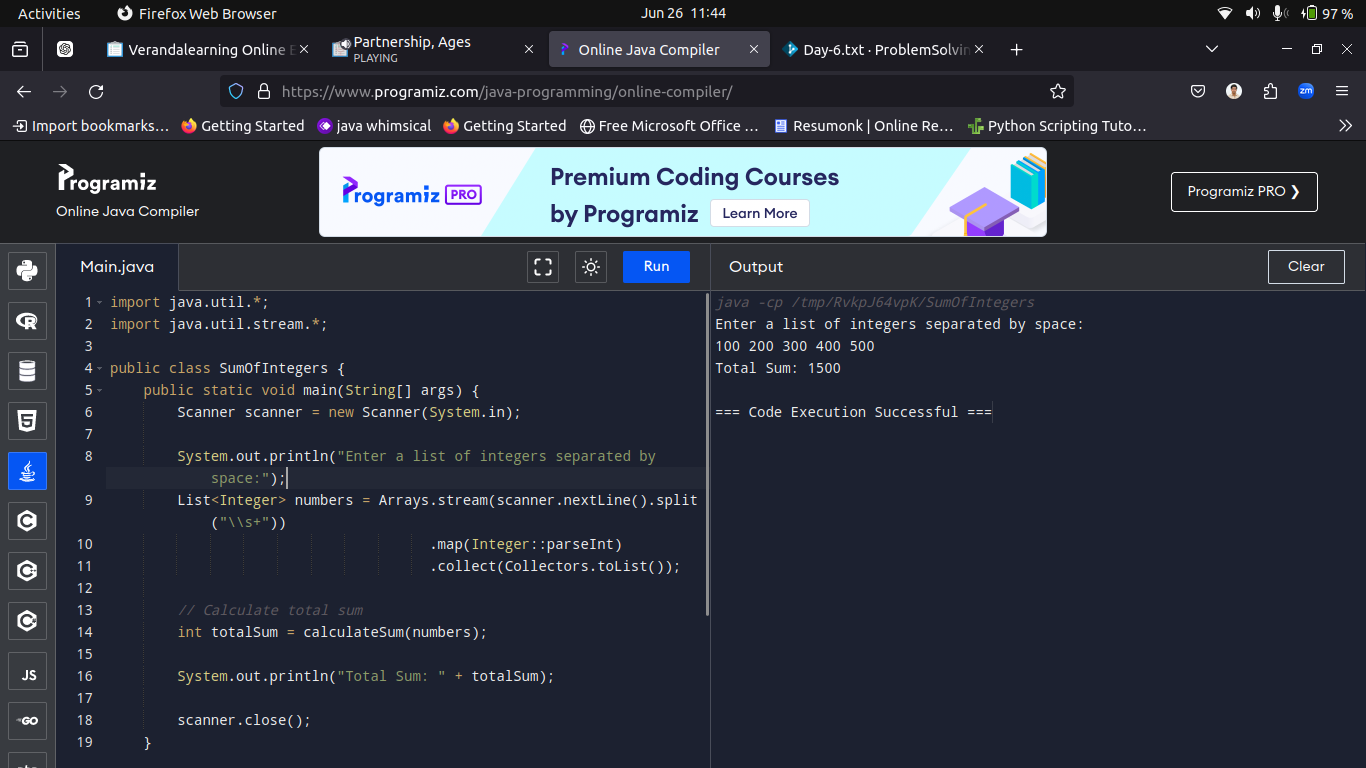
return numbers.stream()

.mapToInt(Integer::intValue)

.sum();

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q2.Filter Even Numbers

Alex is building a utility tool that helps in data analysis. One of the requirements is to filter out even numbers from a list of integers.

Help Alex write a program that reads a list of integers and filters out the even numbers using Java Streams.

Test Case 1:

Input: 1 2 3 4 5 6

Output: Even Numbers: [2, 4, 6]

Test Case 2:

Input: 7 8 9 10 11

Output: Even Numbers: [8, 10]

Solution:

import java.util.\*;

import java.util.stream.\*;

public class EvenNumberFilter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a list of integers separated by space:");

List<Integer> numbers = Arrays.stream(scanner.nextLine().split("\\s+"))

.map(Integer::parseInt)

.collect(Collectors.toList());

// Filter even numbers

List<Integer> evenNumbers = filterEvenNumbers(numbers);

System.out.println("Even Numbers: " + evenNumbers);

}

public static List<Integer> filterEvenNumbers(List<Integer> numbers) {

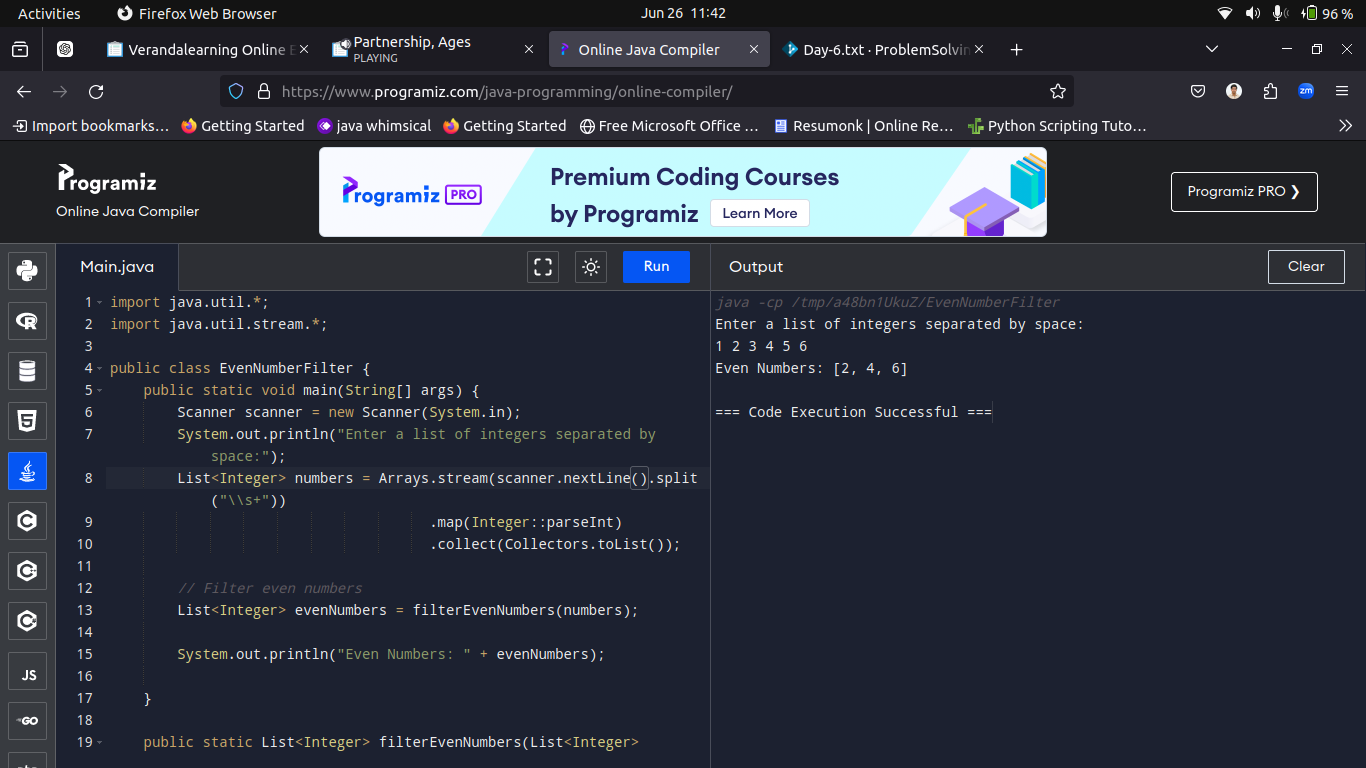
return numbers.stream()

.filter(num -> num % 2 == 0)

.collect(Collectors.toList());

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q3.Olivia is building a text categorization tool that groups words by their length. Help Olivia write a program that reads a list of strings

and groups them by their length using Java Streams.

Test Case 1:

Input: a bb ccc dd eee

Output: Grouped by Length: {1=[a], 2=[bb, dd], 3=[ccc, eee]}

Test Case 2:

Input: apple banana cherry date

Output: Grouped by Length: {5=[apple], 6=[banana, cherry], 4=[date]}

solution:

import java.util.\*;

import java.util.stream.\*;

public class WordGroupByLength {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a list of words separated by space:");

String input = scanner.nextLine();

String[] words = input.split("\\s+");

// Group words by their length

Map<Integer, List<String>> groupedByLength = Arrays.stream(words)

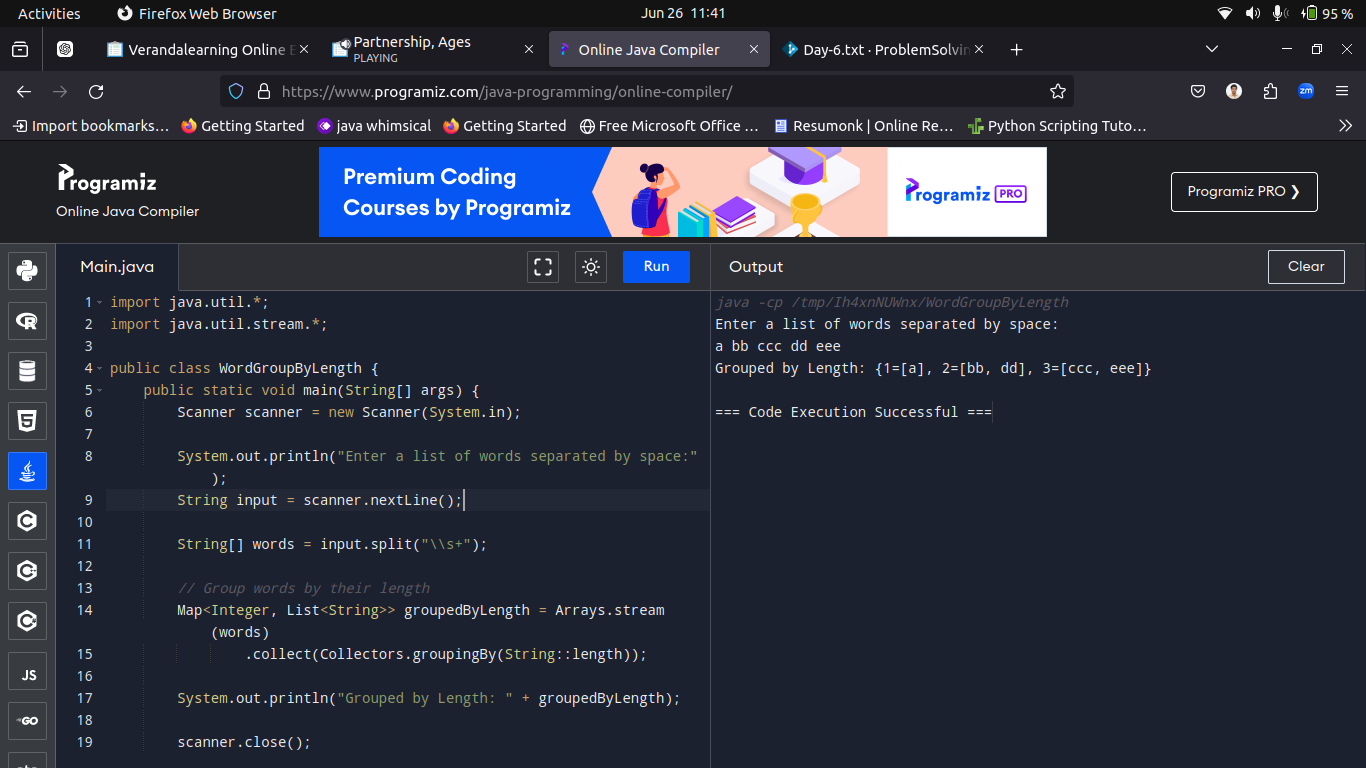
.collect(Collectors.groupingBy(String::length));

System.out.println("Grouped by Length: " + groupedByLength);

scanner.close();

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q4.Lucas is developing a data integration tool that needs to flatten nested lists of strings into a single list. Help Lucas write a program

that reads multiple lists of strings and flattens them into one list using flatMap.

Test Case 1:

Input: a b c|d e f|g h i

Output: Flattened List: [a, b, c, d, e, f, g, h, i]

Test Case 2:

Input: 1 2|3 4 5|6 7 8 9

Output: Flattened List: [1, 2, 3, 4, 5, 6, 7, 8, 9]

Solution:

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

import java.util.\*;

public class FlatteningLists {

public static void main(String[] args) {

// String input = "a b c|d e f|g h i";

Scanner sc = new Scanner(System.in);

String input = sc.nextLine();

// Split the input by '|' to get individual lists of strings

List<String> flattenedList = Arrays.stream(input.split("\\|"))

// Split each list by whitespace to get individual elements

.flatMap(list -> Arrays.stream(list.split("\\s+")))

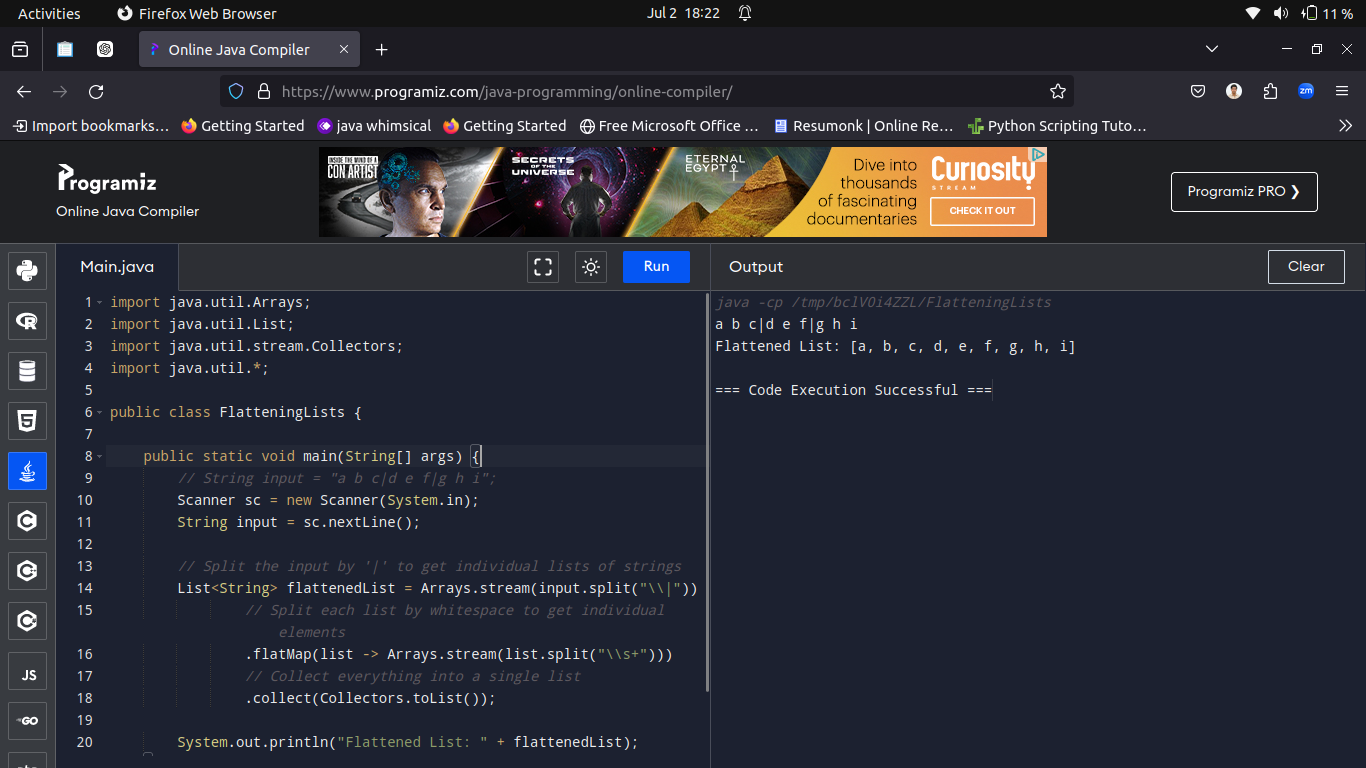
// Collect everything into a single list

.collect(Collectors.toList());

System.out.println("Flattened List: " + flattenedList);

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q5.Bob is creating a tool to check if two given strings are anagrams. Help Bob write a program that reads two strings and checks if

they are anagrams using Java Streams.

Test Case 1:

Input:

String 1: listen

String 2: silent

Output: Are Anagrams: true

Test Case 2:

Input:

String 1: hello

String 2: world

Output: Are Anagrams: false

Solution:

import java.util.\*;

import java.util.stream.\*;

public class AnagramCheckerUsingStreams {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter String 1:");

String str1 = scanner.nextLine();

System.out.println("Enter String 2:");

String str2 = scanner.nextLine();

boolean areAnagrams = checkIfAnagrams(str1, str2);

System.out.println("Are Anagrams: " + areAnagrams);

scanner.close();

}

public static boolean checkIfAnagrams(String str1, String str2) {

// Remove spaces and convert to lower case

String cleanStr1 = str1.replaceAll("\\s", "").toLowerCase();

String cleanStr2 = str2.replaceAll("\\s", "").toLowerCase();

// Convert strings to sorted character streams and compare

return cleanStr1.chars().sorted()

.collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)

.toString()

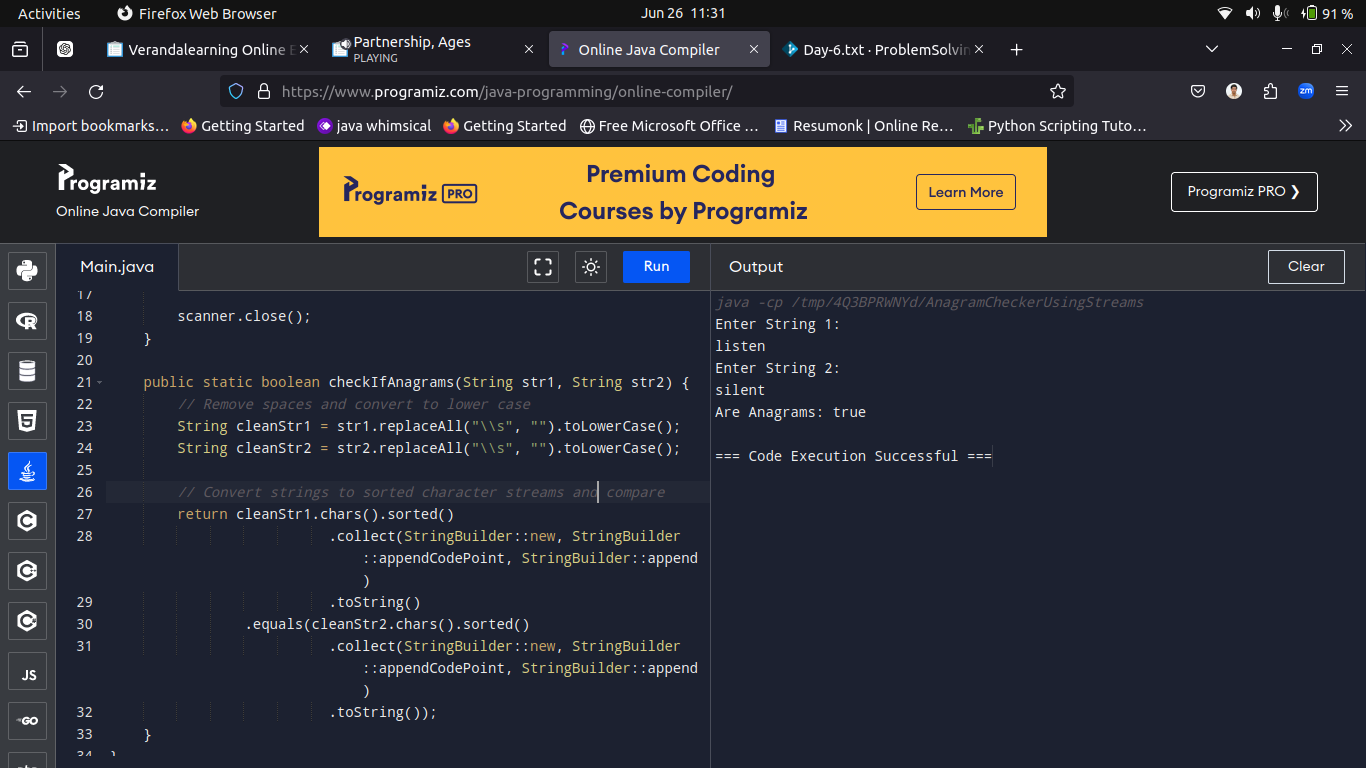
.equals(cleanStr2.chars().sorted()

.collect(StringBuilder::new, StringBuilder::appendCodePoint, StringBuilder::append)

.toString());

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q6.Employee Salary Calculation

Eva works in HR and needs a tool to calculate the total salary of employees with a salary above a certain threshold. Help Eva write a

program that reads employee salaries and a threshold, then calculates the total salary of those above the threshold using Java Streams.

Test Case 1:

Input:

Salaries: 5000 7000 8000 2000

Threshold: 6000

Output: Total Salary above threshold: 15000

Test Case 2:

Input:

Salaries: 3000 4000 5000

Threshold: 4500

Output: Total Salary above threshold: 5000

Code solution:

import java.util.\*;

import java.util.stream.\*;

public class TotalSalaryAboveThreshold {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Read salaries input

System.out.println("Enter employee salaries separated by space:");

List<Integer> salaries = Arrays.stream(scanner.nextLine().split("\\s+"))

.map(Integer::parseInt)

.collect(Collectors.toList());

// Read threshold input

System.out.println("Enter the salary threshold:");

int threshold = scanner.nextInt();

// Calculate total salary above threshold

int totalSalaryAboveThreshold = calculateTotalSalaryAboveThreshold(salaries, threshold);

// Print the result

System.out.println("Total Salary above threshold: " + totalSalaryAboveThreshold);

scanner.close();

}

public static int calculateTotalSalaryAboveThreshold(List<Integer> salaries, int threshold) {

return salaries.stream()

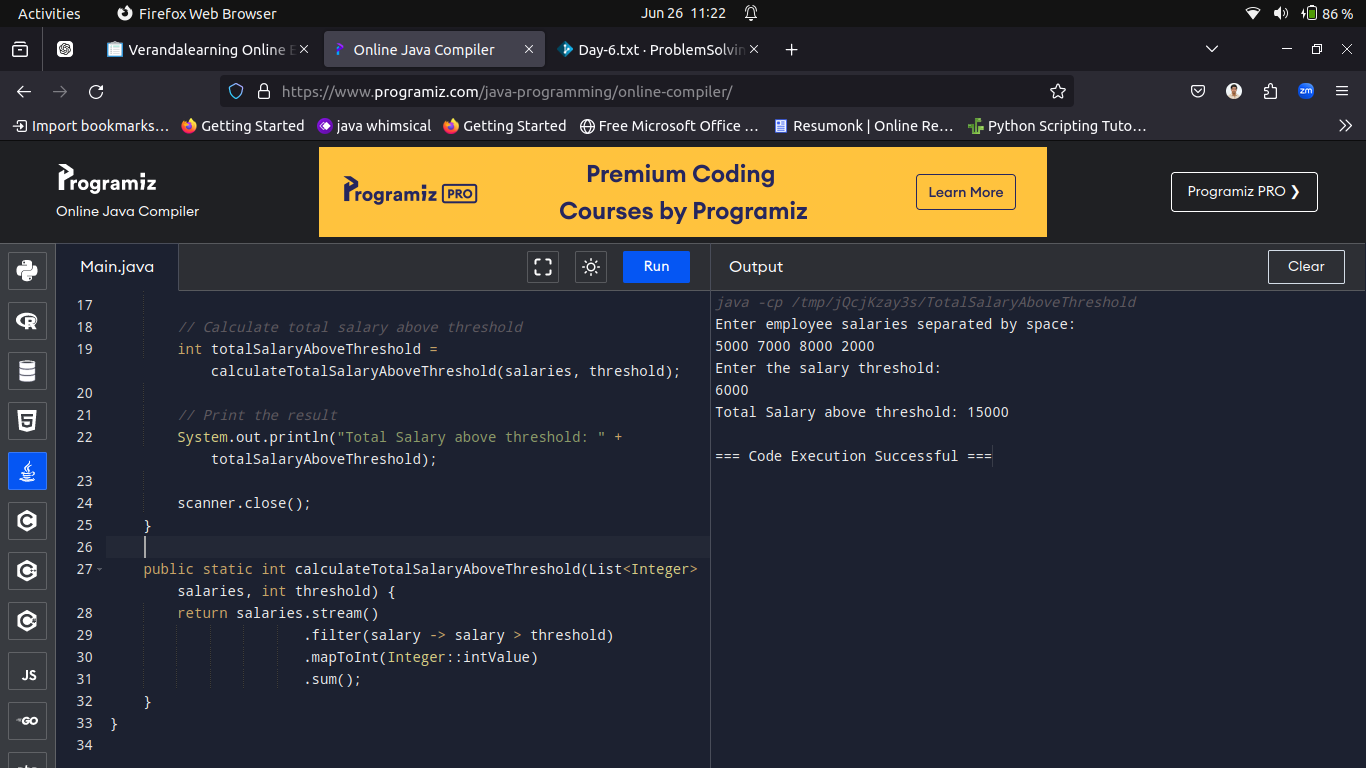
.filter(salary -> salary > threshold)

.mapToInt(Integer::intValue)

.sum();

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q7.Lavish is developing a language learning app and needs to count the number of vowels in a given string.

Help Lavish to write a program that reads a string and counts the vowels using Java Streams.

Test Case 1:

Input: hello world

Output: Number of Vowels: 3

Test Case 2:

Input: programming

Output: Number of Vowels: 3

Solution:

import java.util.Scanner;

public class CountVowelsUsingStreams {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a string:");

String input = scanner.nextLine();

long vowelCount = countVowels(input);

System.out.println("Number of Vowels: " + vowelCount);

}

public static long countVowels(String input) {

// Convert the string to lower case (to handle both upper and lower case vowels)

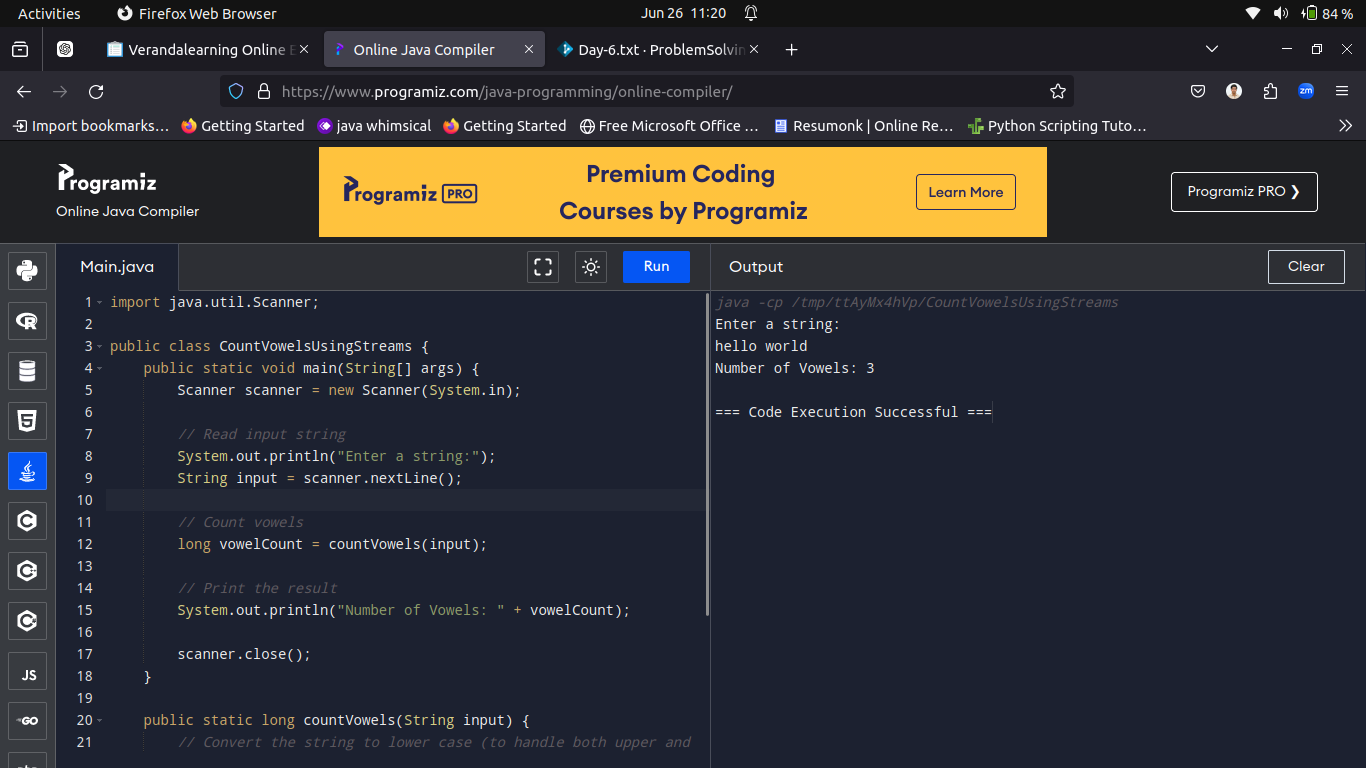
return input.toLowerCase().chars()

.filter(c -> "aeiou".indexOf(c) != -1)

.count();

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q8.Find Duplicates in List

Hannah is working on a data processing tool and needs to find duplicate entries in a list of integers. Help Hannah write a program that

reads a list of integers and identifies the duplicates using Java Streams.

Test Case 1:

Input: 1 2 2 3 4 4 5

Output: Duplicates: [2, 4]

Test Case 2:

Input: 10 20 30 10 40

Output: Duplicates: [10]

Solution:

import java.util.\*;

import java.util.stream.Collectors;

public class FindDuplicatesInList {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Read input list of integers

System.out.println("Enter integers separated by space:");

List<Integer> numbers = Arrays.stream(scanner.nextLine().split("\\s+"))

.map(Integer::parseInt)

.collect(Collectors.toList());

// Find and print duplicates

List<Integer> duplicates = findDuplicates(numbers);

System.out.println("Duplicates: " + duplicates);

scanner.close();

}

public static List<Integer> findDuplicates(List<Integer> numbers) {

// Grouping elements by their occurrences

Map<Integer, Long> countMap = numbers.stream()

.collect(Collectors.groupingBy(

i -> i, Collectors.counting()));

// Filtering elements whose count is greater than 1

List<Integer> duplicates = countMap.entrySet().stream()

.filter(entry -> entry.getValue() > 1)

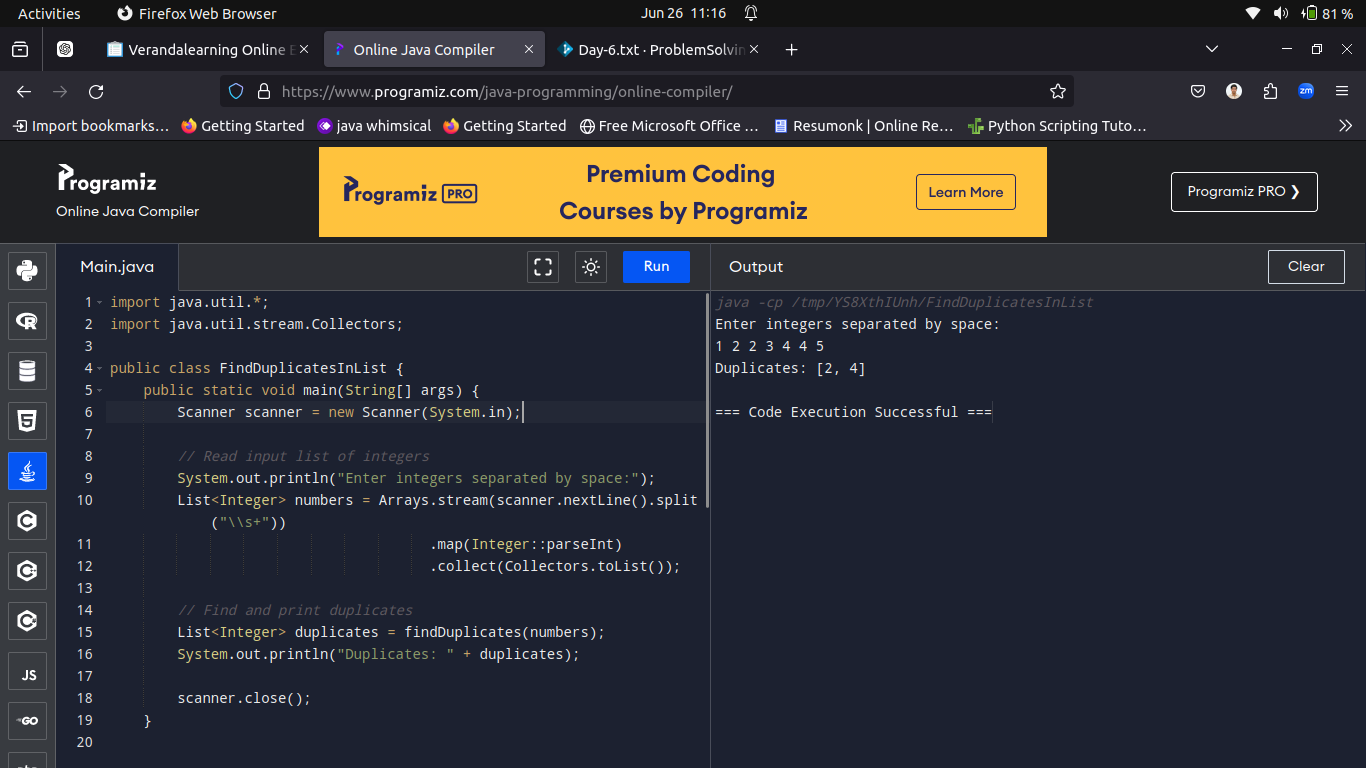
.map(Map.Entry::getKey)

.collect(Collectors.toList());

return duplicates;

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q9.Find Common Elements in Two Lists

Lavish is developing a comparison tool that needs to find common elements between two lists. Help Lavish write a program that reads

two lists of integers and finds the common elements using Java Streams.

Test Case 1:

Input:

List 1: 1 2 3 4 5

List 2: 4 5 6 7 8

Output: Common Elements: [4, 5]

Test Case 2:

Input:

List 1: 10 20 30

List 2: 30 40 50

Output: Common Elements: [30]

Solution:

import java.util.\*;

import java.util.stream.Collectors;

public class CommonElementsInLists {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter elements of List 1 separated by space:");

List<Integer> list1 = Arrays.stream(scanner.nextLine().split("\\s+"))

.map(Integer::parseInt)

.collect(Collectors.toList());

System.out.println("Enter elements of List 2 separated by space:");

List<Integer> list2 = Arrays.stream(scanner.nextLine().split("\\s+"))

.map(Integer::parseInt)

.collect(Collectors.toList());

List<Integer> commonElements = findCommonElements(list1, list2);

System.out.println("Common Elements: " + commonElements);

scanner.close();

}

public static List<Integer> findCommonElements(List<Integer> list1, List<Integer> list2) {

Set<Integer> set2 = new HashSet<>(list2);

List<Integer> commonElements = list1.stream()

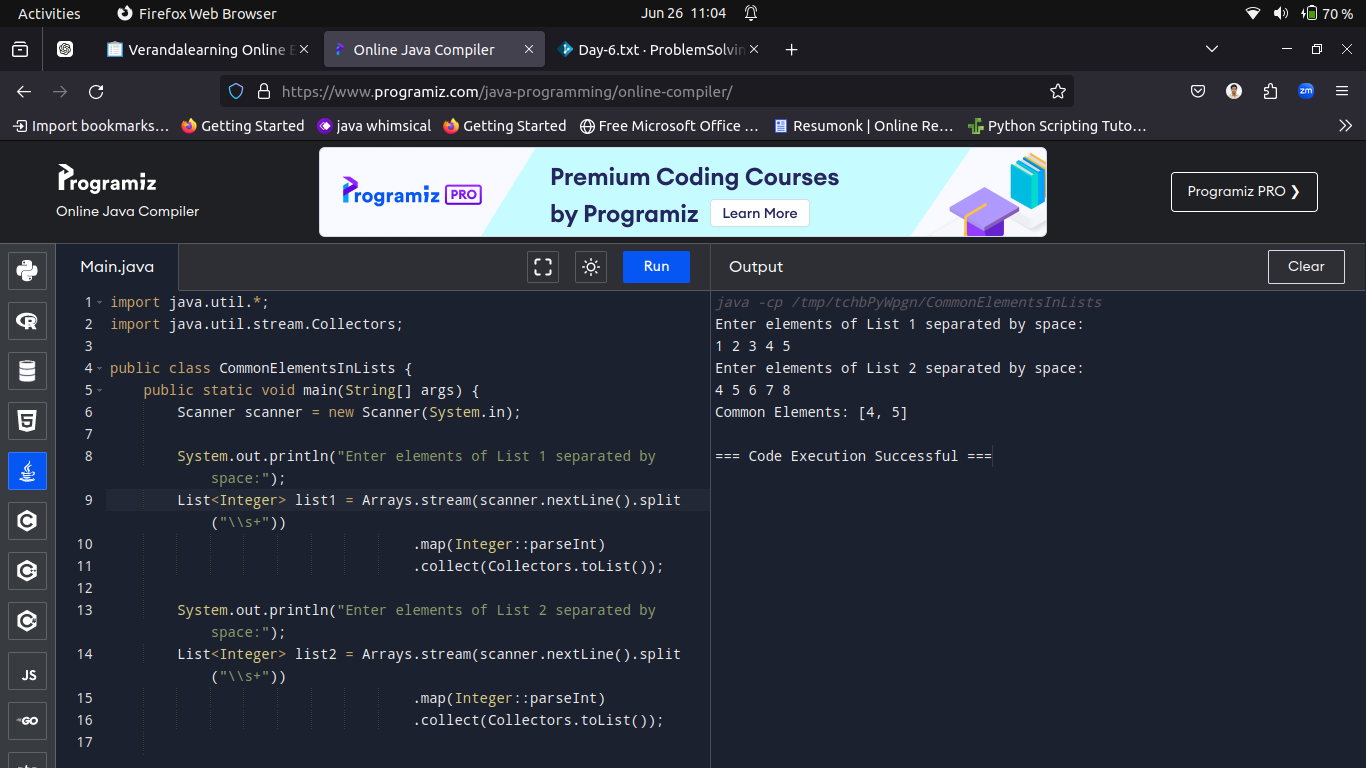
.filter(set2::contains)

.collect(Collectors.toList());

return commonElements;

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------

Q10.Prime Numbers in a Range

Olivia is studying prime numbers and wants to find all prime numbers within a given range. Help Olivia write a program using Java Streams

to generate all prime numbers within the range.

Test Case 1:

Input:

Start: 5

End: 20

Output:

Prime numbers between 5 and 20:

5

7

11

13

17

19

Test Case 2:

Input:

Start: 10

End: 50

Output:

Prime numbers between 10 and 50:

11

13

17

19

23

29

31

37

41

43

47

Solution:

import java.util.stream.IntStream;

import java.util.\*;

public class PrimeNumbersInRange {

public static void main(String[] args) {

// int start = 5;

// int end = 20;

Scanner sc = new Scanner(System.in);

System.out.println("Start:");

int start = sc.nextInt();

System.out.println("End:");

int end = sc.nextInt();

System.out.println("Prime numbers between " + start + " and " + end + ":");

findPrimeNumbersInRange(start, end)

.forEach(System.out::println);

}

public static IntStream findPrimeNumbersInRange(int start, int end) {

return IntStream.rangeClosed(start, end)

.filter(PrimeNumbersInRange::isPrime);

}

public static boolean isPrime(int number) {

if (number <= 1) {

return false;

}

if (number == 2 || number == 3) {

return true;

}

if (number % 2 == 0 || number % 3 == 0) {

return false;

}

int sqrt = (int) Math.sqrt(number) + 1;

for (int i = 6; i <= sqrt; i += 6) {

if (number % (i - 1) == 0 || number % (i + 1) == 0) {

return false;

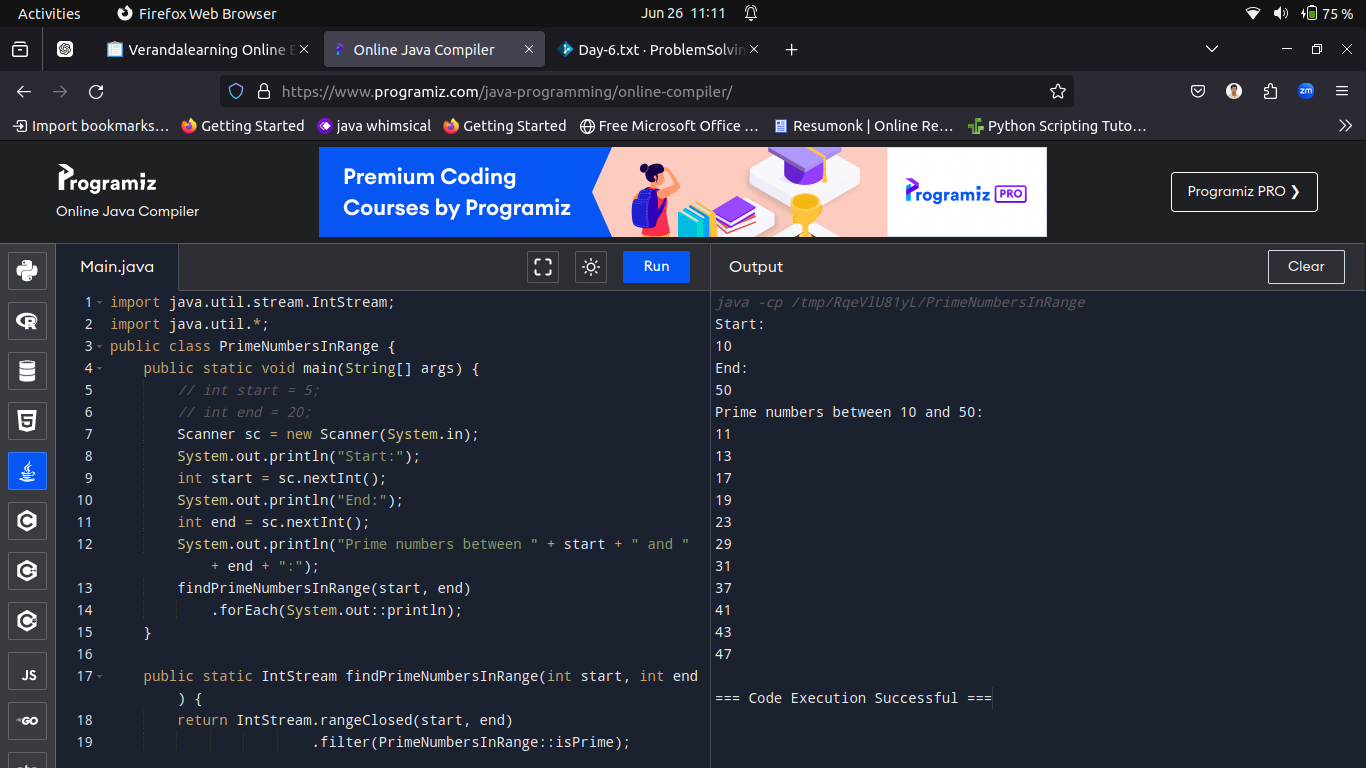
}

}

return true;

}

}



-----------------------------------------------------------------------------------------------------------------------------------------------