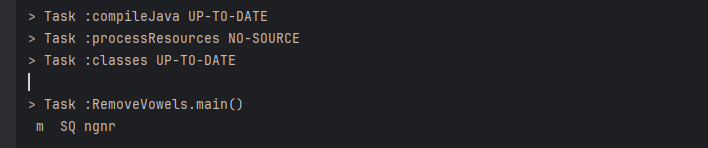
1.Remove vowels code:

public class RemoveVowels {  
 public static void main(String[] args) {  
 String input = "I am a SQA Engineer";  
 String result = *removeVowels*(input);  
 System.*out*.println(result);  
 }  
  
 public static String removeVowels(String input) {  
 return input.replaceAll("[AEIOUaeiou]", "");  
 }  
}

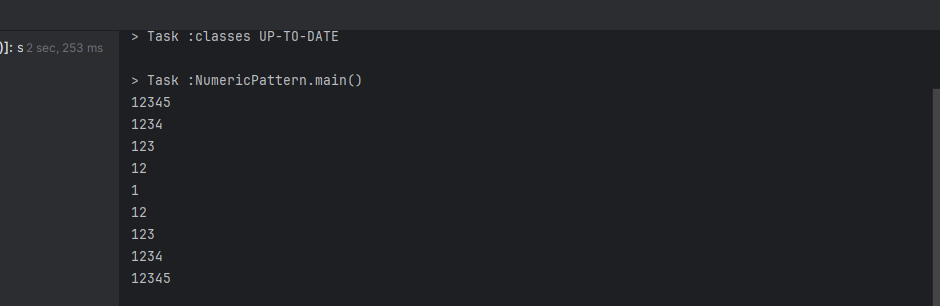
Output:



2.Numeric pattern:

public class NumericPattern {  
 public static void main(String[] args) {  
  
 int maxLength = 5;  
 for (int i = maxLength; i >= 1; i--) {  
 for (int j = 1; j <= i; j++) {  
 System.*out*.print(j);  
 }  
 System.*out*.println();  
 }  
   
 for (int i = 2; i <= maxLength; i++) {  
 for (int j = 1; j <= i; j++) {  
 System.*out*.print(j);  
 }  
 System.*out*.println();  
 }  
 }  
 }

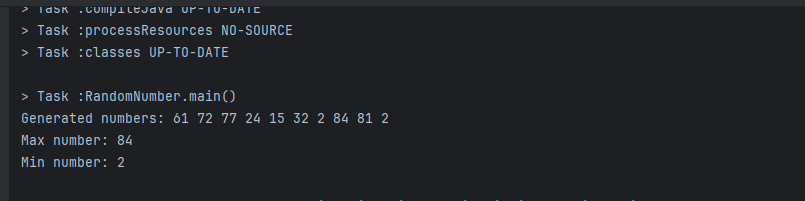
Output:



3.Generate random 10 integer numbers in an array and print out all the numbers from the array and also print the max and min number from the array

import java.util.Random;  
  
public class RandomNumber {  
  
 public static void main(String[] args) {  
 int[] numbers = new int[10];  
 Random random = new Random();  
  
   
 for (int i = 0; i < numbers.length; i++) {  
 numbers[i] = random.nextInt(100); }  
  
   
 System.*out*.print("Generated numbers: ");  
 for (int number : numbers) {  
 System.*out*.print(number + " ");  
 }  
 System.*out*.println();  
  
 int max = numbers[0];  
 int min = numbers[0];  
  
 for (int i = 1; i < numbers.length; i++) {  
 if (numbers[i] > max) {  
 max = numbers[i];  
 }  
 if (numbers[i] < min) {  
 min = numbers[i];  
 }  
 }  
  
   
 System.*out*.println("Max number: " + max);  
 System.*out*.println("Min number: " + min);  
 }  
 }

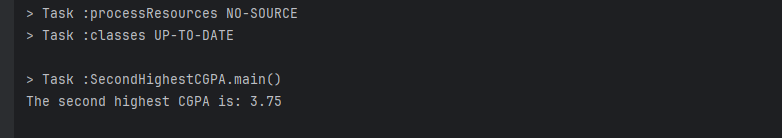
Output:



4.Here is the 10 students CGPA [3.50, 3.52, 3.43, 3.63, 3.48, 3.32, 3.30, 3.60, 3.86, 3.75] Find out who achieved the 2nd highest score. (without sorting, use linear searching algorithm)

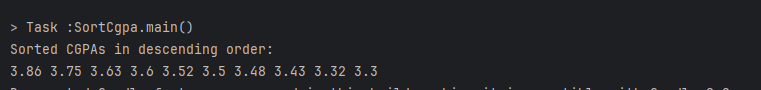
import java.util.Scanner;  
  
public class SearchCgpa {  
  
  
  
 public class BinarySearchCGPA {  
 public static void main(String[] args) {  
  
 double[] cgpas = {3.50, 3.52, 3.43, 3.63, 3.48, 3.32, 3.30, 3.60, 3.86, 3.75};  
  
  
 *selectionSortAscending*(cgpas);  
  
  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter CGPA to search: ");  
 double userCGPA = scanner.nextDouble();  
  
  
 int index = *binarySearch*(cgpas, userCGPA);  
  
  
 if (index != -1) {  
 System.*out*.println("CGPA " + userCGPA + " found at index " + index + ".");  
 } else {  
 System.*out*.println("CGPA " + userCGPA + " not found.");  
 }  
 }  
  
  
 public static void selectionSortAscending(double[] cgpas) {  
 int n = cgpas.length;  
  
  
 for (int i = 0; i < n - 1; i++) {  
 int minIndex = i;  
 for (int j = i + 1; j < n; j++) {  
 if (cgpas[j] < cgpas[minIndex]) {  
 minIndex = j;  
 }  
 }  
  
  
 double temp = cgpas[minIndex];  
 cgpas[minIndex] = cgpas[i];  
 cgpas[i] = temp;  
 }  
 }  
  
  
 public static int binarySearch(double[] cgpas, double target) {  
 int left = 0;  
 int right = cgpas.length - 1;  
  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
  
  
 if (cgpas[mid] == target) {  
 return mid;  
 }  
  
  
 if (cgpas[mid] < target) {  
 left = mid + 1;  
 }  
  
 else {  
 right = mid - 1;  
 }  
 }  
  
  
 return -1;  
 }  
 }  
  
}

Output:



5.Sort the above scores from according to the order of highest CGPA (don’t use Arrays.sort() method, do it programmatically)

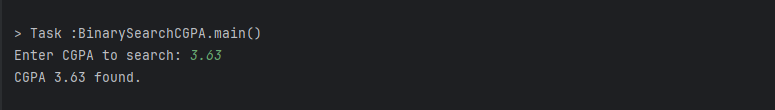
public class SortCgpa {  
 public static void main(String[] args) {  
 double[] cgpas = {3.50, 3.52, 3.43, 3.63, 3.48, 3.32, 3.30, 3.60, 3.86, 3.75};  
 *selectionSortDescending*(cgpas);  
  
 System.*out*.println("Sorted CGPAs in descending order:");  
 for (double cgpa : cgpas) {  
 System.*out*.print(cgpa + " ");  
 }  
 }  
  
 public static void selectionSortDescending(double[] cgpas) {  
 int n = cgpas.length;  
  
 for (int i = 0; i < n - 1; i++) {  
 int maxIndex = i;  
 for (int j = i + 1; j < n; j++) {  
 if (cgpas[j] > cgpas[maxIndex]) {  
 maxIndex = j;  
 }  
 }  
  
  
 double temp = cgpas[maxIndex];  
 cgpas[maxIndex] = cgpas[i];  
 cgpas[i] = temp;  
 }  
 }  
 }

Output: 

6.Take a CGPA as user input. Now from the given array find if your input CGPA is present using binary search algorithm

import java.util.Scanner;  
 public class BinarySearchCGPA {  
 public static void main(String[] args) {  
  
 double[] cgpas = {3.50, 3.52, 3.43, 3.63, 3.48, 3.32, 3.30, 3.60, 3.86, 3.75};  
 *selectionSortAscending*(cgpas);  
  
  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter CGPA to search: ");  
 double userCGPA = scanner.nextDouble();  
  
 int index = *binarySearch*(cgpas, userCGPA);  
  
 if (index != -1) {  
 System.*out*.println("CGPA " + userCGPA + " found. " );  
 } else {  
 System.*out*.println("CGPA " + userCGPA + " not found.");  
 }  
 }  
  
 public static void selectionSortAscending(double[] cgpas) {  
 int n = cgpas.length;  
  
 for (int i = 0; i < n - 1; i++) {  
 int minIndex = i;  
 for (int j = i + 1; j < n; j++) {  
 if (cgpas[j] < cgpas[minIndex]) {  
 minIndex = j;  
 }  
 }  
 double temp = cgpas[minIndex];  
 cgpas[minIndex] = cgpas[i];  
 cgpas[i] = temp;  
 }  
 }  
  
 public static int binarySearch(double[] cgpas, double target) {  
 int left = 0;  
 int right = cgpas.length - 1;  
  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
  
  
 if (cgpas[mid] == target) {  
 return mid;  
 }  
  
  
 if (cgpas[mid] < target) {  
 left = mid + 1;  
 }  
  
 else {  
 right = mid - 1;  
 }  
 }  
  
 return -1;  
 }  
 }

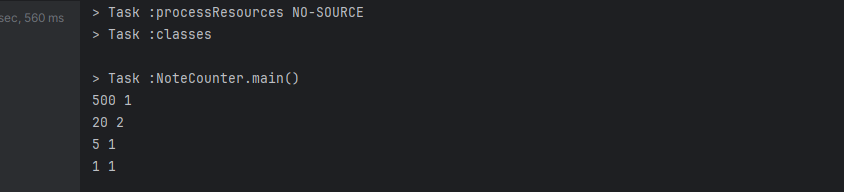
Output:



Write a program that will break down the amount and count notes for any given amount. Here is the notes in the given array: notes=[1000,500,200,100,50,20,10,5,2,1] Marks:10 Input: 546 Output: 500 1 20 2 5 1 1 1

import java.util.LinkedHashMap;  
import java.util.Map;  
  
public class NoteCounter {  
 public static void main(String[] args) {  
 int amount = 546;  
 int[] notes = {1000, 500, 200, 100, 50, 20, 10, 5, 2, 1};  
  
 Map<Integer, Integer> noteCount = *countNotes*(amount, notes);  
  
 *printNotes*(noteCount);  
 }  
  
 public static Map<Integer, Integer> countNotes(int amount, int[] notes) {  
 Map<Integer, Integer> noteCount = new LinkedHashMap<>();  
  
 for (int note : notes) {  
 if (amount >= note) {  
 int count = amount / note;  
 amount -= note \* count;  
 noteCount.put(note, count);  
 }  
 }  
  
 return noteCount;  
 }  
  
 public static void printNotes(Map<Integer, Integer> noteCount) {  
 for (Map.Entry<Integer, Integer> entry : noteCount.entrySet()) {  
 System.*out*.println(entry.getKey() + " " + entry.getValue());  
 }  
 }  
}

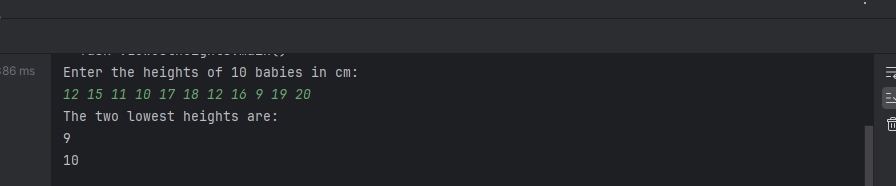
Output:



Take input as height of 10 babies in cm. Now find out the 2 lowest height of babies. (Don’t use Arrays.sort() method)

import java.util.Scanner;  
  
 public class LowestHeights {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int[] heights = new int[10];  
  
  
 System.*out*.println("Enter the heights of 10 babies in cm:");  
 for (int i = 0; i < 10; i++) {  
 heights[i] = scanner.nextInt();  
 }  
  
  
 int lowest = Integer.*MAX\_VALUE*;  
 int secondLowest = Integer.*MAX\_VALUE*;  
  
 for (int height : heights) {  
 if (height < lowest) {  
 secondLowest = lowest;  
 lowest = height;  
 } else if (height < secondLowest && height != lowest) {  
 secondLowest = height;  
 }  
 }  
  
  
 System.*out*.println("The two lowest heights are:");  
 System.*out*.println(lowest);  
 System.*out*.println(secondLowest);  
  
 scanner.close();  
 }  
 }

Output:



Count number of words, number of characters without spaces, number of vowels and consonant from the given string Input: I am a SQA Engineer Output: Words: 5 Chars: 15 Vowel: 8 Consonant: 7

import java.util.Scanner;  
  
public class StringAnalysis {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
   
 System.*out*.println("Enter a string:");  
 String input = scanner.nextLine();  
  
  
 int wordCount = *countWords*(input);  
 int charCount = *countCharactersWithoutSpaces*(input);  
 int vowelCount = *countVowels*(input);  
 int consonantCount = *countConsonants*(input);  
  
  
 System.*out*.println("Words: " + wordCount);  
 System.*out*.println("Chars: " + charCount);  
 System.*out*.println("Vowels: " + vowelCount);  
 System.*out*.println("Consonants: " + consonantCount);  
  
 scanner.close();  
 }  
  
 public static int countWords(String input) {  
 if (input == null || input.isEmpty()) {  
 return 0;  
 }  
 String[] words = input.split("\\s+");  
 return words.length;  
 }  
  
 public static int countCharactersWithoutSpaces(String input) {  
 if (input == null || input.isEmpty()) {  
 return 0;  
 }  
 input = input.replace(" ", "");  
 return input.length();  
 }  
  
 public static int countVowels(String input) {  
 int count = 0;  
 input = input.toLowerCase();  
 for (char c : input.toCharArray()) {  
 if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {  
 count++;  
 }  
 }  
 return count;  
 }  
  
 public static int countConsonants(String input) {  
 int count = 0;  
 input = input.toLowerCase();  
 for (char c : input.toCharArray()) {  
 if (c >= 'a' && c <= 'z' && !(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')) {  
 count++;  
 }  
 }  
 return count;  
 }  
}

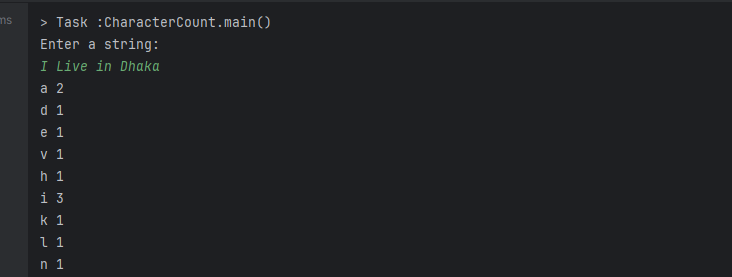
Output:



Find the number of occurrences of characters in a String Marks:5 String: I live in Dhaka Output: i 3 l 1 v 1 e 1 d 1 h 1 a 2 k 1

import java.util.HashMap;  
import java.util.Map;  
import java.util.Scanner;  
  
public class CharacterCount {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 System.*out*.println("Enter a string:");  
 String input = scanner.nextLine();  
  
  
 Map<Character, Integer> charCount = *countCharacters*(input);  
  
  
 for (Map.Entry<Character, Integer> entry : charCount.entrySet()) {  
 System.*out*.println(entry.getKey() + " " + entry.getValue());  
 }  
  
 scanner.close();  
 }  
  
 public static Map<Character, Integer> countCharacters(String input) {  
 Map<Character, Integer> charCount = new HashMap<>();  
  
  
 input = input.toLowerCase().replace(" ", "");  
  
  
 for (char c : input.toCharArray()) {  
 charCount.put(c, charCount.getOrDefault(c, 0) + 1);  
 }  
  
 return charCount;  
 }  
}

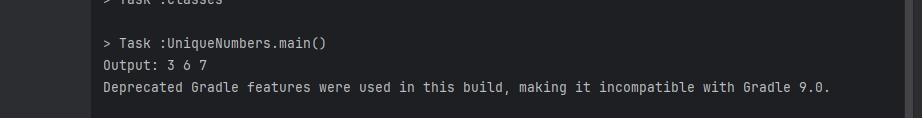
Output:



Print the numbers which are not duplicate from the given array. Marks:5 numbers=[1, 1, 2, 3, 1, 2, 4, 5, 6, 5, 4, 7] Output: 3, 6, 7

import java.util.HashMap;  
import java.util.Map;  
  
public class UniqueNumbers {  
 public static void main(String[] args) {  
 int[] numbers = {1, 1, 2, 3, 1, 2, 4, 5, 6, 5, 4, 7};  
  
  
 Map<Integer, Integer> numberCount = *countNumbers*(numbers);  
  
  
 System.*out*.print("Output: ");  
 for (Map.Entry<Integer, Integer> entry : numberCount.entrySet()) {  
 if (entry.getValue() == 1) {  
 System.*out*.print(entry.getKey() + " ");  
 }  
 }  
 }  
  
 public static Map<Integer, Integer> countNumbers(int[] numbers) {  
 Map<Integer, Integer> numberCount = new HashMap<>();  
  
   
 for (int number : numbers) {  
 numberCount.put(number, numberCount.getOrDefault(number, 0) + 1);  
 }  
  
 return numberCount;  
 }  
}

Output:



A core i 7 laptop price is 85000 tk and a gaming mouse price is 2500 tk. If I buy the laptop and 1 piece mouse, what will be my total cost after giving 15% discount? [Extract the digits from the paragraph and calculate the price]

import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class CalculateDiscountedPrice {  
 public static void main(String[] args) {  
 String paragraph = "A core i 7 laptop price is 85000 tk and a gaming mouse price is 2500 tk. If I buy the laptop and 1 piece mouse, what will be my total cost after giving 15% discount?";  
  
  
 int laptopPrice = *extractNumber*(paragraph, "85000");  
 int mousePrice = *extractNumber*(paragraph, "2500");  
 int discountPercentage = *extractNumber*(paragraph, "15");  
  
  
 int totalCost = laptopPrice + mousePrice;  
  
  
 double discount = (totalCost \* discountPercentage) / 100.0;  
 double finalCost = totalCost - discount;  
  
  
 System.*out*.printf("Total cost after %d%% discount: %.2f tk%n", discountPercentage, finalCost);  
 }  
  
 public static int extractNumber(String text, String defaultValue) {  
 Pattern pattern = Pattern.*compile*("\\d+");  
 Matcher matcher = pattern.matcher(text);  
 while (matcher.find()) {  
 if (matcher.group().equals(defaultValue)) {  
 return Integer.*parseInt*(matcher.group());  
 }  
 }  
 return 0;   
 }  
}

